US006356965B1

(12) United States Patent
Broyles et al.

(10) **Patent No.:** **US 6,356,965 B1**
(45) **Date of Patent:** **Mar. 12, 2002**

(54) **HOTKEY FOR NETWORK SERVICE BOOT**

(75) Inventors: **Paul J. Broyles**, Cypress; **Don R. James, Jr.**, Houston, both of TX (US)

(73) Assignee: **Compaq Computer Corporation**, Houston, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/149,336**

(22) Filed: **Sep. 8, 1998**

(51) **Int. Cl.**^{7} **G06F 13/14**; G06F 13/20

(52) **U.S. Cl.** **710/104**; 713/2; 709/220; 709/226; 714/23

(58) **Field of Search** 713/1, 2; 714/23; 710/104; 709/220–226

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,274,816	A	*	12/1993	Oka	713/2
5,430,845	A	*	7/1995	Rimmer et al.	709/301
5,448,741	A	*	9/1995	Oka	713/2
5,579,522	A	*	11/1996	Christeson et al.	713/2
5,596,711	A	*	1/1997	Burckhartt et al.	714/23
5,694,600	A	*	12/1997	Khenson et al.	713/2
5,696,968	A	*	12/1997	Merkin	713/2
5,715,456	A	*	2/1998	Bennett et al.	713/2
5,822,582	A	*	10/1998	Doragh et al.	713/2
5,854,905	A	*	12/1998	Garney	710/104
5,935,242	A	*	8/1999	Madany et al.	713/1
6,079,016	A	*	6/2000	Park	713/2
6,101,601	A	*	8/2000	Matthews et al.	713/2

FOREIGN PATENT DOCUMENTS

JP 8-179937 A * 7/1996

OTHER PUBLICATIONS

Network Working Group, RFC 1350, *The TFTP Protocol* (Revision 2), Jul. 1992.

Network Working Group, RFC 1541, *Dynamic Host Configuration Protocol*, Oct. 1993.

Network PC System Design Guidelines, *A Reference for Designing Net PC Systems for Use with the Microsoft® Windows® and Windows NT® Operating Systems*, Version 1.0b, Aug. 5, 1997.

Wired for Management Baseline, Version 1.1a, *Specification to Help Reduce TCO for Business PCs*, Aug. 28, 1997.

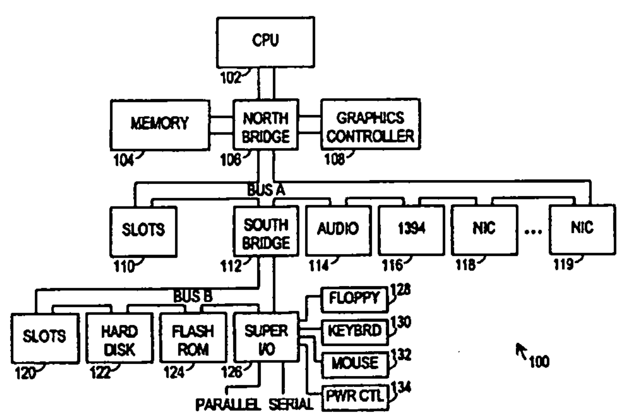
* cited by examiner

Primary Examiner—Thomas Lee
Assistant Examiner—Rehana Perveen
(74) *Attorney, Agent, or Firm*—Conley, Rose & Tayon, P.C.; Michael F. Heim; Daniel J. Krueger

(57) **ABSTRACT**

A computer system is provided with a dynamically reconfigurable boot order. In one embodiment, the computer comprises a user input device, a nonvolatile memory, a network interface, a boot trigger, and a CPU. The CPU is coupled to the user input device to detect a predetermined key press, coupled to the boot trigger to detect the assertion of a system reset signal, and coupled to the nonvolatile memory to retrieve a system BIOS in response to assertion of the system reset signal. The CPU executes the BIOS to initialize the computer system, and as part of the system initialization, the CPU determines a first target boot-up device. Preferably if the predetermined key has been pressed during the system initialization, the CPU alters the default boot order to select the network interface as the first target boot up device. The network interface is configurable to retrieve an operating system from a network device for the CPU to execute. The disclosed embodiment advantageously provides for reduced system installation and maintenance effort, and thereby lead to reduced costs for owners of computer networks.

13 Claims, 3 Drawing Sheets



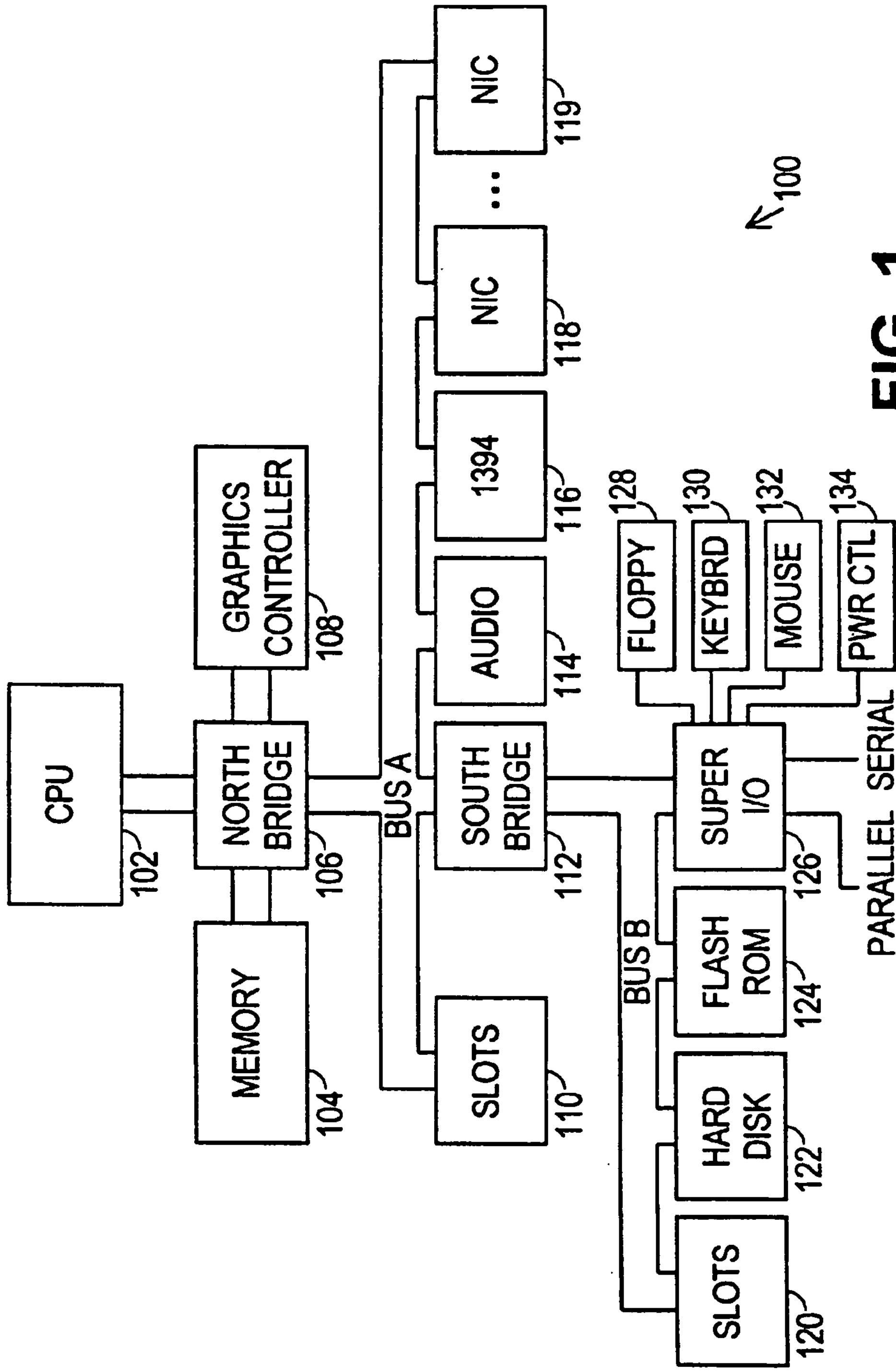

CPU
102
NORTH BRIDGE
106
MEMORY
104
GRAPHICS CONTROLLER
108
BUS A
SOUTH BRIDGE
112
SLOTS
110
AUDIO
114
1394
116
NIC
118
NIC
119
100
BUS B
SLOTS
120
HARD DISK
122
FLASH ROM
124
SUPER I/O
126
FLOPPY
128
KEYBRD
130
MOUSE
132
PWR CTL
134
PARALLEL
SERIAL

FIG. 1

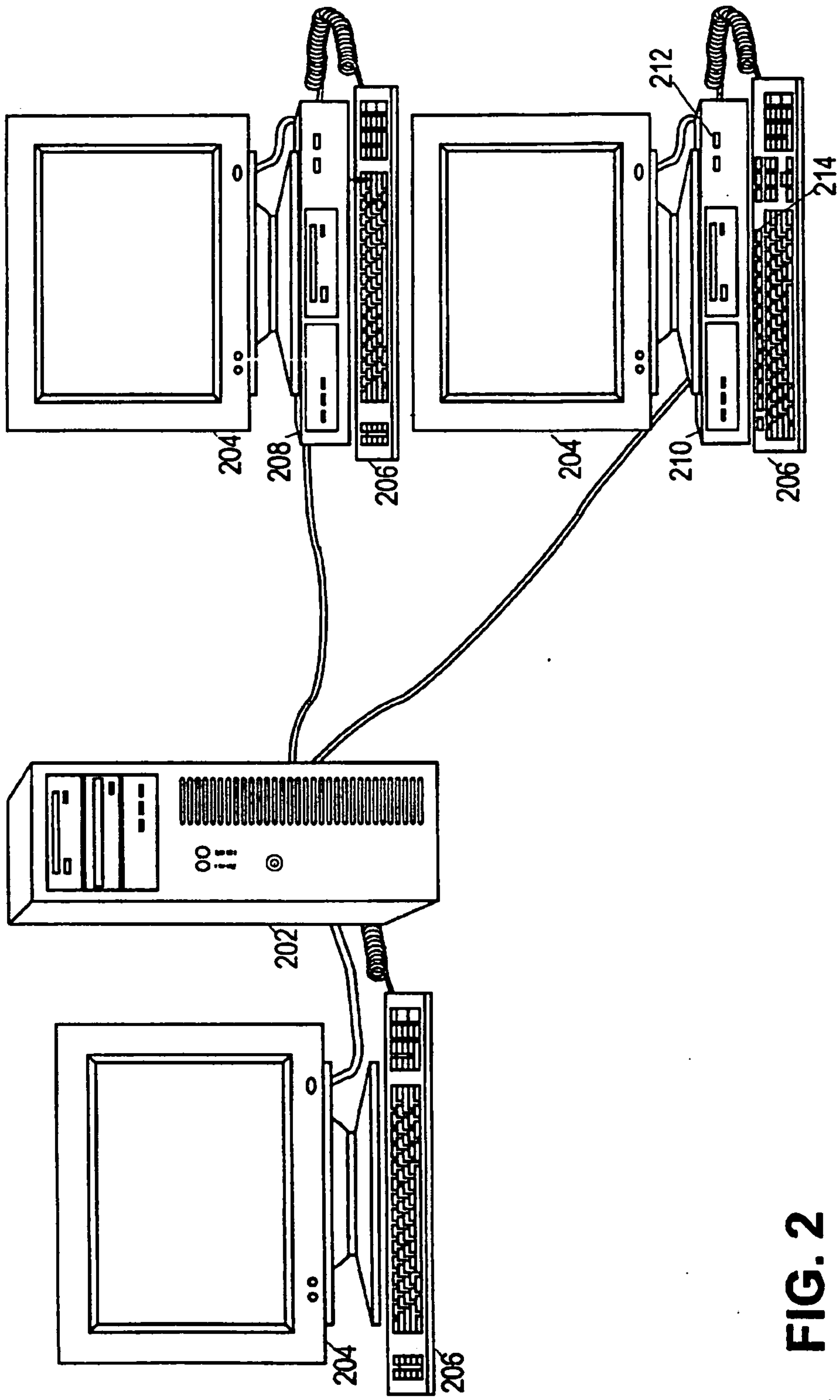


FIG. 2

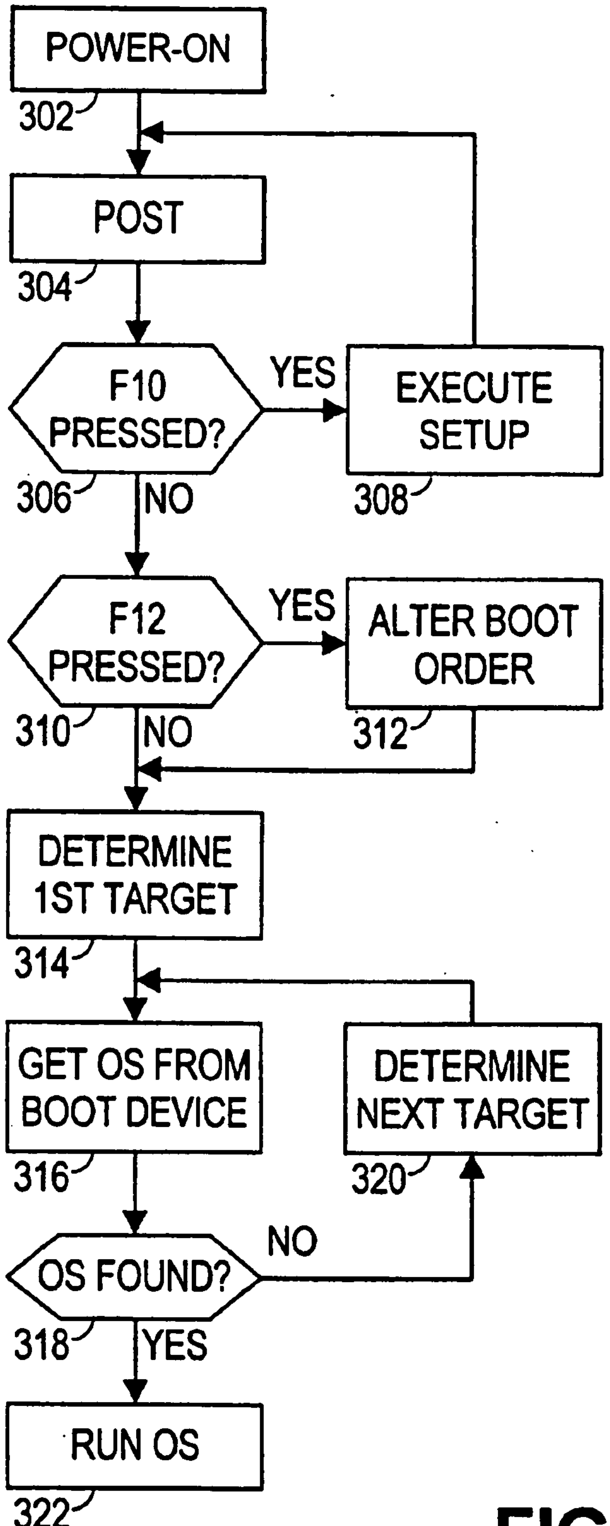

POWER-ON
302
POST
304
F10 PRESSED?
306
YES
NO
EXECUTE SETUP
308
F12 PRESSED?
310
YES
NO
ALTER BOOT ORDER
312
DETERMINE 1ST TARGET
314
GET OS FROM BOOT DEVICE
316
DETERMINE NEXT TARGET
320
OS FOUND?
318
NO
YES
RUN OS
322

FIG. 3

HOTKEY FOR NETWORK SERVICE BOOT

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to a system and method for altering the order in which boot devices are tried during system initialization. More particularly, the present invention relates to a computer system having a boot order that is adjusted if a hotkey is pressed during the initialization of the system.

2. Background of the Invention

Due to the advent of power management technology and the more recent "instant-on" efforts, there are many ways in which a computer may exist in the "OFF" state. Examples include hard off (power is disconnected), soft off (power is supplied only to components which monitor activity external to the system), suspend mode (contents of memory are stored on disk and current state of computer is preserved while power consumption is reduced to a minimum level), and sleep mode (the clock signal is reduced or halted to some or all of the system components during periods of inactivity). The sleep and suspend modes may each be invoked at various levels, depending on the particular implementation of these modes, and recovery from these modes is implementation specific.

Turning a computer "ON" from the hard-off or the soft-off state causes the computer to begin an initialization process (often referred to as "boot-up"). In the initialization process, a system reset signal is asserted and released. After the de-assertion of the reset signal, many of the system peripheral components initialize themselves, retrieve configuration information from dedicated electrically erasable programmable read-only memories (EEPROMs), and enter an initialized state. At the same time, the central processing unit (CPU) resets itself and searches for instructions on how to prepare the system for operation. The initial instructions typically are included in the system's basic input/output system (system BIOS) which is executable code stored in a nonvolatile memory such as a read-only memory (ROM). The BIOS is built-in software that contains low-level code to control and provide access to the keyboard, display screen, disk drives, serial communications, and a number of miscellaneous functions. The BIOS also specifies a boot-up sequence for the CPU to execute to make the computer ready for operation. The CPU normally begins executing initialization routines from the BIOS ROM, but subsequently copies the BIOS code to main memory from which the BIOS code may thereafter be executed during normal computer operations.

Typically, the first thing that the BIOS instructs the CPU to do in the boot up process is to perform what is called the Power-On Self-Test, or POST for short. The POST is a built-in diagnostic program that checks much of the computer's hardware to ensure that everything is present and functioning properly before the BIOS begins the actual initialization process. Some additional tests are performed later in the boot process. If any fatal errors are encountered, the boot process stops. After the initial POST, the BIOS instructs the CPU to locate the video system's built in BIOS program and to execute it to initialize the video system. The CPU then displays the BIOS's startup screen, and searches for other devices to see if any of them have initialization routines. If any other device initialization routines (e.g. IDE hard drive) are found, they are executed as well.

After the video system has been initialized, the CPU will normally display a prompt similar to "Press F10 to enter

SETUP" and continue booting up the system. If the user properly responds to the prompt, e.g. by pressing the F10 key within 3 seconds of the appearance of the prompt, the CPU will halt the system boot-up sequence and execute a BIOS setup program where the user is able to view and modify system configuration parameters. After the user exits the BIOS setup program, the computer restarts the boot-up sequence so that any modified configuration parameters may be properly taken into account during system initialization.

The CPU does more tests on the system, including the memory count-up test which may be viewed on the video display. If an error is encountered after the initialization of the video system, a text error message will generally be displayed on the video display. The BIOS boot-up sequence includes a "system inventory" of sorts, performing more tests to determine what sort of hardware is in the system. Modem BIOSes have many automatic settings and may, among other things, automatically determine memory timing based on what kind of memory it finds installed in the computer. Many BIOSes can also dynamically set hard drive parameters and access modes, and will determine these at roughly this time during the initialization process. The BIOS will also now instruct the CPU to search for and label logical devices (COM and LPT ports). If the BIOS supports the Plug and Play standard, the CPU will detect and configure Plug and Play devices at this time and display a message on the screen for each one it finds. The CPU will often display a summary screen about the system configuration and begin a search for a boot device.

Some modem BIOSes contain a boot table that specifies the order of devices from which the system should try to boot. If a first target device in the list is present and properly configured for booting, the BIOS will boot the system from this device. If the target device that the system tries is not found, the CPU will then try the next device in the boot table, and continue until it finds a bootable device. If no boot device at all can be found, the system will normally display an error message and then freeze up the system.

After having identified a target boot drive, the BIOS instructs the CPU to look for boot information to start the operating system boot process. For example, with a hard disk, the CPU may search for a master boot record at cylinder 0, head 0, sector 1 (the first sector on the disk). If the CPU finds the master boot record, the CPU starts the process of booting the operating system, using the information in the boot sector. At this point, the code indicated by the boot sector takes over from the BIOS code.

The boot devices which may be accessed during the above boot-up sequence include any nonvolatile storage device. Floppy disks, hard disks, magnetic tape, CD-ROMs, Flash ROMs, and network server disks are all examples of devices which can serve as a boot device. To be a boot device, a device should hold a copy of an operating system or application which is to be executed after system initialization. Often the boot device includes a "boot-sector" that informs the CPU of the operating system's exact storage location. Local devices (i.e. devices included in the computer or directly connected to the computer) may in some systems be preferred over remote devices (i.e. devices that need to be accessed via a network or shared communications link) for booting a computer system, while in other systems remote devices may be preferred.

Many variations exist for the boot-up sequence conducted by the BIOS. As computer hardware has become increasingly reliable, proposals have been made to eliminate POST tests altogether from the normal boot-up sequence. For

example, in "Simple Boot Flag Specification: Version 1.0", Microsoft has proposed the use of a register to communicate boot options to the system BIOS. The boot flags are PNPOS, BOOTING, and DIAG. The PNPOS flag is asserted if the operating system normally used by the computer is Plug-and-Play capable. If this is the case, the BIOS does not need to spend time configuring components that the operating system will configure. The DIAG flag is de-asserted if hardware tests are considered unnecessary. In this case, the BIOS can skip the POST. The BOOTING flag, if asserted, indicates that the previous boot attempt did not successfully complete and the BIOS may choose to ignore the other flags and provide a complete system test and configuration sequence.

When a computer is in a soft-off, suspend, or sleep state, it may be configured to awaken if activity is detected, e.g. movement of a mouse or detection of a "wake up" data packet by a network interface card. The source of the triggering activity may come from a local mechanism (i.e. a switch or sensor of any kind such as a power switch, a reset switch, a pressable key, a pressure sensor, a mouse, a joystick, a touch pad, a microphone, or a motion sensor), or the trigger source may be remote.

Under certain circumstances, it would be desirable to be able to easily reconfigure the order in which devices are targeted for booting the computer. For example, a system administrator wishing to install a new operating system might prefer to bypass the local drives and boot directly from a network drive. In conventional systems, this would require the following steps by the administrator:

1. Reboot the computer.
2. Enter the BIOS setup program.
3. Find and change the boot order option.
4. Reboot the computer from the network.
5. Login to the network and install the new operating system.
6. Reboot the computer.
7. Enter the BIOS setup program.
8. Find and change to boot order option.
9. Reboot the computer normally.

This procedure is cumbersome and requires a significant amount of the administrator's time. Since this procedure must be repeated by the administrator for each computer in the network, a more streamlined approach is desirable.

SUMMARY OF THE INVENTION

Accordingly, there is provided herein a computer system having a dynamically reconfigurable boot order. In one embodiment, the computer comprises a user input device, a nonvolatile memory, a network interface, a boot trigger, and a CPU. The CPU is coupled to the user input device to detect a predetermined key press, coupled to the boot trigger to detect the to assertion of a system reset signal, and coupled to the nonvolatile memory to retrieve a system BIOS in response to assertion of the system reset signal. The CPU executes the BIOS to initialize the computer system, and as part of the system initialization, the CPU determines a first target boot-up device. Preferably if the predetermined key has been pressed during the system initialization, the CPU alters the default boot order to select the network interface as the first target boot up device. The network interface is configurable to retrieve an operating system from a network device for the CPU to execute. The disclosed embodiment advantageously provides for reduced system installation and maintenance effort, and thereby lead to reduced costs for owners of computer networks.

BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:

FIG. 1 is a block diagram of a computer system illustrating various system components;

FIG. 2 is an illustrative diagram of a simple computer network; and

FIG. 3 illustrates a method for dynamically altering a computer's boot order.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the appended claims.

In addition, certain terms are used throughout the following description and claims to refer to particular system components. This document does not intend to distinguish between components that differ in name but not function. In the following discussion and in the claims, the terms "including" and "comprising" are used in an open-ended fashion, and thus should be interpreted to mean "including, but not limited to . . . ". Also, the term "couple" or "couples" is intended to mean either an indirect or direct electrical connection. Thus, if a first device couples to a second device, that connection may be through a direct electrical connection or through an indirect electrical connection via other devices and connections.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Turning now to the figures, FIG. 1 illustrates an example of a configuration of various computer components that may be found in a computer system. It is noted that many other representative configurations exist and that this embodiment is described for illustrative purposes. The computer system **100** is preferably provided with a hotkey (shown in FIG. 2) for dynamically reconfiguring the boot order. If the hotkey is pressed during system initialization, the computer system **100** is configured to bypass local boot devices and boot from a remote boot device. Remote boot procedures using a Dynamic Host Configuration Protocol (DHCP—published as RFC 1541) and Trivial File Transfer Protocol (TFTP—published as RFC 1350) are described in "Network PC System Design Guidelines: Version 1.0b—Aug. 5, 1997" by Compaq, Dell, Hewlett Packard, Intel, and Microsoft, and "Wired for Management Baseline: Version 1.1a" published Aug. 28, 1997 by Intel. The foregoing references are hereby incorporated by reference.

The computer system **100** of FIG. 1 includes a CPU **102** coupled to a bridge logic device **106** via a CPU bus. The bridge logic device **106** is sometimes referred to as a "North bridge" for no other reason than it often is depicted at the upper end of a computer system drawing. The North bridge **106** also couples to a main memory array **104** by a memory bus, and may further couple to a graphics controller **108** via an accelerated graphics port (AGP) bus. The North bridge **106** couples CPU **102**, memory **104**, and graphics controller **108** to the other peripheral devices in the system through a

primary expansion bus (BUS A) which may be implemented as a peripheral component interconnect (PCI) bus or an extended industry standard architecture (EISA) bus. Various components that comply with the communications protocol and electrical requirements of BUS A may reside on this bus, such as an audio device 114, a IEEE 1394 interface device 116, and a network interface card (NIC) 118. The system may include more than one network interface, as indicated by NIC 119. These components may be integrated onto the motherboard or they may be plugged into expansion slots 110 that are connected to BUS A.

If other secondary expansion buses are provided in the computer system, as is typically the case, another bridge logic device 112 is used to couple the primary expansion bus (BUS A) to the secondary expansion bus (BUS B). This bridge logic 112 is sometimes referred to as a "South bridge" reflecting its location vis-a-vis the North bridge 106 in a typical computer system drawing. An example of such bridge logic is described in U.S. Pat. No. 5,634,073, assigned to Compaq Computer Corporation. Various components that comply with the communications protocol and electrical requirements of BUS B may reside on this bus, such as hard disk controller 122, Flash ROM 124, and Super I/O controller 126. Slots 120 may also be provided for plug-in components that comply with the protocol of BUS B. Flash ROM 224 stores the system BIOS that is executed by CPU 202 during system initialization.

The Super-Input/Output (Super I/O) controller 126 typically interfaces to basic input/output devices such as a keyboard 130, a mouse 132, a floppy disk drive 128, a parallel port, a serial port, and sometimes a power controller 134 and various other input switches such as a power switch and a suspend switch. The Super I/O controller 126 preferably has the capability to handle power management functions such as reducing or terminating power to components such as the floppy drive 130, and blocking the clock signals that drive components such as the bridge devices 106, 112 thereby inducing a sleep mode in the expansion buses. The Super I/O controller 126 may further assert System Management Interrupt (SMI) signals to various devices such as the CPU 102 and North bridge 106 to indicate special conditions pertaining to input/output activities such as sleep mode.

Super I/O controller 126 may include battery-backed CMOS memory for storing BIOS configuration parameters for system 100, and may further include a counter or a Real Time Clock (RTC). The RTC may be used to track the activities of certain components such as the hard disk 122 and the primary expansion bus, so that controller 126 can induce a sleep mode or reduced power mode after a predetermined time of inactivity. The Super I/O controller 126 may also induce a low-power suspend mode if the suspend switch is pressed, in which the power is completely shut off to all but a few selected devices. Exempted devices might be the Super I/O controller 126 itself and NIC 118.

When a computer is in a soft-off, suspend, or sleep state, the Super I/O controller 126 may be configured to rouse the computer if activity is detected, e.g. a power switch closure, movement of the mouse 132 or detection of a "wake up" data packet by NIC 118. The source of the triggering activity may come from a local mechanism (i.e. a switch or sensor of any kind such as a power switch, a reset switch, a pressable key, a pressure sensor, a mouse, a joystick, a touch pad, a microphone, a motion sensor, or a biometric device (e.g. fingerprint reader)), or the trigger source may be remote and perhaps communicated to the computer system by a network, serial bus, modem, or some other communications link. When the computer system 100 is in a soft-off or hard-off state, and the Super I/O controller 126 detects a power switch closure, controller 126 asserts a system reset signal and initiates system boot-up. During system boot-up, the CPU 102 retrieves the BIOS from Flash ROM 124 and executes the BIOS. The BIOS stores various system configuration parameters in CMOS memory, and retrieves these parameters to initialize and configure various system components to place the system in readiness for operation by a user. One of the configuration parameters may be a preferred boot-order selected by the user in the BIOS setup program.

Referring still to FIG. 1, the BIOS stored in Flash ROM 124 includes a boot table which specifies a default boot order. The default boot order is the order in which the boot devices are normally accessed in an effort to locate and execute an operating system. The user can normally alter the default boot order by changing a parameter in CMOS memory contained in the Super I/O controller 126. One example of a popular boot order is:

CD-ROM
FLOPPY DRIVE
HARD DISK #1
SCSI
NETWORK DRIVE #1
NETWORK DRIVE #2

This order is popular because it allows the computer to operate from a local drive (e.g. the CD-ROM, floppy drive, or hard disk) whenever possible, thereby maximizing the performance of most systems. However, computers which for some reason may have missing or corrupted operating systems can still boot up using another copy of an operating system that may be stored elsewhere (e.g. the master copy of the operating system stored on a network server).

FIG. 2 shows an example of a computer network in which a central server 202 is coupled to a first computer 208 and a second computer 210. Central server 202 and computers 208 and 210 may each be equipped with a terminal 204 and an input device 206, and may each be provided with an architecture similar to that of FIG. 1. Computer 210 is shown having a power button 212 and a function key 214. Most modem keyboards have ten or twelve function keys labeled F1-F10 or F1-F12 which are often located in a single row along the upper edge of the keyboard or in a group on the left-hand side of the keyboard. These function keys are often assigned to special purpose functions by software applications. For example, pressing F1 while a software application is running commonly opens up a "help window" whereby a user may find usage instructions for the software application.

Any key or key-combination (multiple keys which are pressed simultaneously) which invokes a special purpose function (i.e. a function other than causing an alphanumeric character to appear on the display) when pressed is commonly referred to as a "hotkey". One standard hotkey that is used for re-booting a computer is Control-Alt-Delete, in which the three keys Control, Alt, and Delete are pressed simultaneously. As the computer is booting up, a hotkey may be used by the user to enter the BIOS setup program. Typical examples of hotkeys used to invoke the setup program include Control-S, Shift-F1, Alt-F2, or F10. The BIOS software will normally display a prompt message indicating the key or key-combination which should be used by the user to gain access to the BIOS setup program.

It can be appreciated that the system administrator might desire to perform system maintenance of computers in the network without working through the cumbersome nine-step

procedure outlined in the background section. It may be further appreciated that the system administrator might wish to boot each of the computers from a master copy of the operating system as part of the system maintenance. Examples of when this could be desirable include: installing a new operating system, virus scanning, and executing automated maintenance software. Providing a hot key for temporarily reconfiguring the boot order substantially reduces the effort required by the system administrator, as evidenced by the following example using F12 as the hot key, although any other unassigned key or key-combination may also be used:

1. Reboot the computer.
2. Press the F12 key to boot from the network.
3. Login to the network and install the new operating system.
4. Reboot the computer normally.

Some operating systems will allow the last two steps to be automated so that all the system administrator must do is reboot and press F12.

FIG. 3 shows an illustrative flowchart which may be implemented by a computer's BIOS to provide dynamic configuration of the boot order. The flowchart includes a power-on step **302**, a POST step **304**, a first hot-key test step **306**, a Setup execution step **308**, a second hot-key test step **310**, a boot order reconfiguration step **312**, a determine first target step **314**, an Operating System load step **316**, a load success test **318**, a determine next target step **320**, and an execute Operating System step **322**. In the flowchart of FIG. 3, the power-on step **302** can have multiple triggers, including a press of the power button **212** and detection of a wake-up packet by NIC **118**. In the power-on step **302**, power is provided to the computer's various system components including the CPU **102**. The CPU **102** retrieves the BIOS from Flash ROM **124** and begins the POST portion of the boot-up sequence in step **304**. During the POST step **304**, the CPU **102** preferably displays a message on the display terminal similar to "F10=Setup, F12=Network boot". This message may be displayed after the CPU **102** has initialized the PCI devices and determined if indeed there is a network interface present.

In test step **306**, the CPU **102** determines if the Setup hotkey has been pressed. If so, the CPU **102** executes the BIOS setup program in step **308**. Otherwise, in step **310** the CPU **102** determines if the Network Boot hotkey has been pressed. To determine when keys are pressed, the CPU may employ polling or interrupt operations. A CPU which performs polling regularly checks the keyboard to determine if a key is being pressed, while a CPU which employs interrupts halts any current operations to respond to an interrupt signal generated by the keyboard when a key is pressed.

If so the CPU **102** determines that the Network Boot hotkey has been pressed, the CPU **102** alters the boot order in step **312**. In a first embodiment, the CPU **102** temporarily re-orders the entries in the boot table so that the network boot devices are accessed first and the local drives are only accessed if the computer fails to boot from the network. However, the boot order parameter in CMOS memory and the original table stored in Flash ROM **124** preferably are not altered so that subsequent computer boot-ups will follow the original boot order. In a second embodiment, the CPU **102** bypasses the local drives and only attempts to boot from network devices in the boot table. In a third embodiment, the CPU **102** simply creates a new boot table entry for a network device as a temporary first entry before other entries in the existing boot table. This new entry may not necessarily be entered into the existing table; rather, the variable used to retrieve entry values from the table is simply initialized with the settings of a network boot device. Since this may lead to effectively having two entries for the network device, the CPU **102** may also implement a flag to indicate that the network drive has already been attempted to prevent a second attempt if the network drive appears again later in the boot table.

Using the original boot order, or the new boot order if a hotkey is detected, the CPU **102** determines a first target boot device in step **314** and attempts to retrieve an operating system from the target boot device in step **316**. In step **318**, the CPU **102** determines if the retrieval attempt was successful. If so, then in step **322**, the BIOS turns control over to the retrieved operating system. If not, then in step **320**, the CPU **102** determines a next target boot device, and loops back to step **316**. The loop is repeated until an operating system is located or all boot devices have been unsuccessfully tried. If all boot devices have been unsuccessfully tried, the computer declares a fatal error and ceases all attempts to boot up.

An appendix is provided with assembler code excerpts from one BIOS implementation to illustrate one method for dynamically reconfiguring a computer system's boot order. After the POST has completed, the BIOS makes a call to the AttemptBypassBBS procedure. This procedure checks to determine if the F12 key has been pressed and, if so, calls the AttemptRIPL procedure. The AttemptRIPL procedure scans in order through the boot table for network interface cards (NICs) and boot-entry vector devices (BEV devices), and for each one found, calls the pbbsAttemptBoot procedure. The PCI Plug-N-Play standard defines BEV devices. A BEV device may be a network interface that includes an option ROM with executable code for locating a network server, retrieving a boot image of the operating system from the network server, and initiating execution of the boot image. The pbbsAttemptBoot procedure determines if the selected device is a bootable NIC or BEV device and attempts to boot from those devices determined to be bootable by calling a PXENVBoot procedure or a pbbsBootBEV procedure, respectively. If a successful boot occurs, no return from these procedures is expected.

Accordingly, a system and method which provide a dynamically reconfigurable boot order for a computer have been described. Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. For example, the default boot order may be configured so that faster boot devices are tried before slower boot devices. In particular, to reduce boot-up time a local hard drive may be tried before a floppy disk or CD-ROM. A hotkey may be provided to reorder the boot-up sequence so that booting from the removable media devices (e.g. floppy disks, CD-ROMs) is tried before resorting to the local hard drive. It is intended that the following claims be interpreted to embrace all such variations and modifications.

APPENDIX

The following code excerpts are taken from a BIOS implementation of a hotkey for dynamically reconfiguring a computer's boot order to bypass local boot devices and boot from a network.

```
bbsTableEntry struc                             ; BOOTSTRAP TABLE ENTRY
    DeviceType dw           EMPTY_TYPE          ; DEVICE TYPE
        EMPTY_TYPE          EQU 0               ; define types of devices
        FLOPPY_TYPE         EQU 1               ;
        HDISK_TYPE          EQU 2               ;
        CD_TYPE             EQU 3               ;
        PCMCIA_TYPE         EQU 4               ;
        USB_TYPE            EQU 5               ;
        NIC_TYPE            EQU 6               ;
        BEV_TYPE            EQU 80h             ;
        UNKNOWN_TYPE        EQU 0FFh            ;
StatusRec record R1:4=0,MediaPresent:2,BootFail:1,EnableBoot:1,R2:4=0,LastBoot:4
    StatusFlags StatusRec <>                    ; STATUS FLAGS
        NO_BOOT_MEDIA       EQU 0               ; StatusRec.MediaPresent equates
        UNKNOWN_MEDIA       EQU 1               ;
        BOOT_MEDIA          EQU 2               ;
    union                                       ; BOOT HANDLER LOCATION
      struc                                     ;
        BootHandler  dd     7C000000h           ; Segment:Offset of Boot handler:PT925
      ends                                      ;
      struc                                     ;
        BootHandlerOff dw  ?                    ;
        BootHandlerSeg dw  ?                    ;
      ends                                      ;
    ends                                        ;
    union                                       ; DEVICE DESCRIPTION LOCATION
      struc                                     ;
        String DescString dd 0F0000000h;        Segment:Offset of Descript. String:PT925
      ends                                      ;
      struc                                     ;
        DescStringOff dw    ?                   ;
        DescStringSeg dw    ?                   ;
      ends                                      ;
    ends                                        ;
    Expansion       dd      0                   ; RESERVED
bbsTableEntry ends                              ;
;*************************************************************
;
;       rbbsIPLTable - IPL Table Data Structure
;
;*************************************************************
rbbsIPLTable bbsTableEntry MAX_DEVICES DUP ({ }) ; IPL table
;*************************************************************
;
;   AttemptBootDiag - Check for signature if found then boot    ;PT925
;
;*************************************************************
AttemptBypassBBS  proc  near
    call rkbdF12Pressed?                        ; Q: User pressed F12?
    jz    short   @f                            ;    N: skip to next text
    call AttemptRIPL                            ;    Y: boot to network
@@:
.
.
.
AttemptBypassBBS endp
;*************************************************************
;
;   AttemptRIPL
;
;   This function attempts to boot from the highest priority NIC_TYPE or
;   BEV_TYPE device
;
;   Procedure:
;       Scan through IPL priority until a NIC_TYPE or BEV_TYPE is found
;       call pbbsAttempBoot with index
;
;*************************************************************
public AttemptRIPL
AttemptRIPL proc near
        push    ax
        push    cx
        push    si
        push    es
```

-continued

APPENDIX
The following code excerpts are taken from a BIOS implementation of a hotkey for dynamically reconfiguring a computer's boot order to bypass local boot devices and boot from a network.

```
        mov     ax,SEG rbbsIPLDevs              ;
        mov     es,ax                           ;
        mov     ch,cs:[rbbsIPLDevs]             ; Initialize loop max
        xor     cl,cl                           ; start with first device
RIPLLoop:                                       ;
        lea     si,rbbsIPLTable                 ; SI points to first entry in IPL tbl
        call    pbbsGetTblIndex                 ; Get index into table
        movzx   ax,bl                           ; AX = index into IPL table
        mov     di,SIZE bbsTableEntry           ; DI=size of IPL entry
        mul     di                              ; AX=DI * index
        add     si,ax                           ; SI -> bbsTableEntry[bl]
        cmp     es: [si].DeviceType,NIC_TYPE    ;Q: Onboard NIC device!
        je      short GotIt                     ;     Y: We're done
        cmp     es[si].DeviceType,BEV_TYPE      ; Q: BEV Device?
        je      short GotIt                     ;     Y: We're done
        inc     cl                              ; go to next device
        cmp     cl,ch                           ; Q: Last device?
        jb      short RIPLLoop                  ;     N: keep looking
        jmp     RIPL_Done                       ;     Y: get out
GotIt:
        call    pbbsAttemptBoot                 ; BL has index of device to boot
RIPL_Done:
        pop     es
        pop     si
        pop     cx
        pop     ax
        ret
AttemptRIPL endp
;************************************************************
;
;   pbbsAttemptBoot - Attempt to boot from specified IPL device
;
;   Entry: BL - index into IPL table
;   Return (if it returns): CL, CH, BH restored
;
;************************************************************
pbbsAttemptBoot proc near                       ;
.
.
.
@@:                                             ;
        cmp     ax,NIC_TYPE                     ;
        jne     SHORT @F                        ;
        mov     ax,NIC_TYPE                     ;
        call    pbbsBootMsg                     ;
        call    pbbsBootUpNIC?                  ; Q: Up or down NIC
        call    PXENVBoot                       ; Boot NIC
        jmp     SHORT no_boot                   ; Recover
@@:
        cmp     ax,BEV_TYPE                     ;
        jne     SHORT no_boot                   ; Hmmmmmmm
        test    cs[si].StatusFlags,0200h        ; Q: First attempt?
        jz      BEV_OK                          ;     Y: go boot
        call    rbbsBEVFailedMsg                ;     N: no boot after failed
        jmp     no_boot
BEV_OK:
        call    pbbsBootMsg                     ;
        test    cs[si].StatusFlags,0100h        ; Q: Ignore this BEV?
        jz      no_boot                         ;     Y: Exit
        call    pbbsBootBEV                     ; Boot BEV
.
.
.pbbsAttemptBoot endp
```

What is claimed is:

1. A computer system having a dynamically reconfigurable boot order, wherein the computer system comprises:

a user input device;

a nonvolatile memory that stores a BIOS which specifies a first boot order;

a boot trigger configurable to momentarily assert a system reset signal;

a CPU coupled to the boot trigger to detect the assertion of the system reset signal, coupled to the nonvolatile memory to retrieve the BIOS in response to assertion of the system reset signal, and coupled to the user input device to detect one or more key presses,

wherein the CPU determines a second boot order if the predetermined key press is detected, wherein the second boot order is a re-ordering of the first boot order.

2. The computer system of claim 1, wherein the user input device is a keyboard.

3. The computer system of claim 1, wherein the predetermined key press is part of a predetermined key-combination.

4. The computer system of claim 1, wherein the predetermined key press is operation of a function key.

5. The computer system of claim 1, wherein the predetermined key press is operation of a function key labeled F12.

6. The computer system of claim 1, wherein the boot trigger is a power switch.

7. A computer system having a dynamically reconfigurable boot order, wherein the computer system comprises:

a user input device;

a nonvolatile memory that stores a BIOS;

a network interface configurable to retrieve an initial program via a network

a boot trigger configurable to momentarily assert a system reset signal;

a CPU coupled to the user input device to detect a predetermined key press, coupled to the boot trigger to detect the assertion of the system reset signal, coupled to the nonvolatile memory to retrieve the BIOS in response to assertion of the system reset signal, and coupled to the network interface to receive the initial program,

wherein the CPU is configured to execute the BIOS to determine a first target boot-up device, wherein the CPU determines that the first target boot-up device is the network interface if the predetermined key press is detected, and wherein the CPU determines that the first target boot-up device is a local device otherwise,

wherein the BIOS specifies a first boot order, and wherein the CPU determines a second boot order if the predetermined key press is detected, wherein the second boot order is a re-ordering of the first boot order such that the second boot order begins with a network interface present in the first boot order.

8. A method for booting up a computer, wherein the method comprises: detecting a trigger event;

applying power to a CPU;

retrieving a BIOS that specifies a first boot order;

determining if a predetermined key has been pressed; and

creating a second boot order different from the first boot order if the predetermined key has been pressed after the trigger event, wherein the second boot order is a re-ordering of the first boot order.

9. The method of claim 8, wherein the second boot order begins with a network interface.

10. The method of claim 8, wherein the predetermined key is a function key.

11. The method of claim 8, wherein the predetermined key press is part of a predetermined key-combination.

12. A method for booting up a computer, wherein the method comprises:

detecting a trigger event;

applying power to a CPU;

retrieving a BIOS that specifies a first boot order;

determining if a predetermined key has been pressed;

if the predetermined key has not been pressed:

selecting a first target boot device from the first boot order; otherwise,

if the predetermined key has been pressed:

creating a second boot order which is a reordering of the first boot order;

selecting a first target boot device from the second boot order; and

accessing the first target boot device to retrieve an operating system.

13. The method of claim 12, further comprising:

determining if the operating system has been found;

selecting a next target boot device if the operating system has not been found; and

executing the operating system if the operating system has been found.

* * * * *

US006317828B1

(12) **United States Patent**
Nunn

(10) **Patent No.: US 6,317,828 B1**
(45) **Date of Patent: Nov. 13, 2001**

(54) **BIOS/UTILITY SETUP DISPLAY**

(75) Inventor: **Susan Nunn**, Austin, TX (US)

(73) Assignee: **Dell USA, L.P.**, Round Rock, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/191,833**

(22) Filed: **Nov. 13, 1998**

(51) **Int. Cl.**[7] **G06F 15/177**
(52) **U.S. Cl.** **713/2**
(58) **Field of Search** 713/1, 2; 710/1, 710/8–15

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,361,358	*	11/1994	Cox et al.	713/2
5,504,905	*	4/1996	Cleary et al.	713/1
5,696,968		12/1997	Merkin	395/652
5,727,213		3/1998	Vander Kamp et al.	395/681
5,860,001	*	1/1999	Cromer et al.	713/1
5,999,989	*	12/1999	Patel	710/1

OTHER PUBLICATIONS

IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practices, IEEE Std 1275–1994, Mar. 17, 1994, pp. 1–43.*

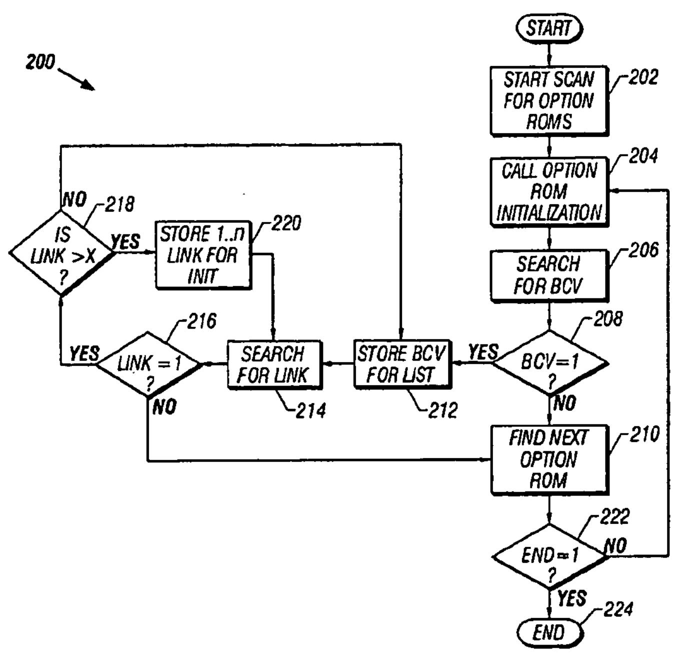
* cited by examiner

Primary Examiner—Thomas Black
Assistant Examiner—Mary Wang
(74) *Attorney, Agent, or Firm*—Skjerven Morrill MacPherson LLP; Ken J. Koestner; Margaret M. Kelton

(57) **ABSTRACT**

A method and computer system for a system setup program includes a routine for displaying a subset of bootable devices from an adapter card, such as a PERC card, on a system from which a user may specify a bootable device to serve as a boot drive, and bootable devices to serve as alternative boot drives. The computer system includes a system Basic Input/Output System (BIOS), a system processor, a system memory coupled to the system processor, at least one expansion slot coupled to the system processor via a bus, at least one adapter coupled to the computer system via the at least one expansion slot, and a system BIOS ROM code. The BIOS ROM code detects a plurality of bootable devices on the computer system, selects from the plurality of bootable devices a preselected number of bootable devices for display, and displays the subset of bootable devices on a setup display. The setup display allows a user to specify a bootable device to serve as a boot drive of the computer system. The method includes detecting a plurality of bootable devices on the computer system, selecting from the plurality of bootable devices a preselected number of bootable devices for display, and displaying the subset of bootable devices on a setup display. The setup display allows a user to specify a bootable device to serve as a boot drive from the computer system.

31 Claims, 2 Drawing Sheets



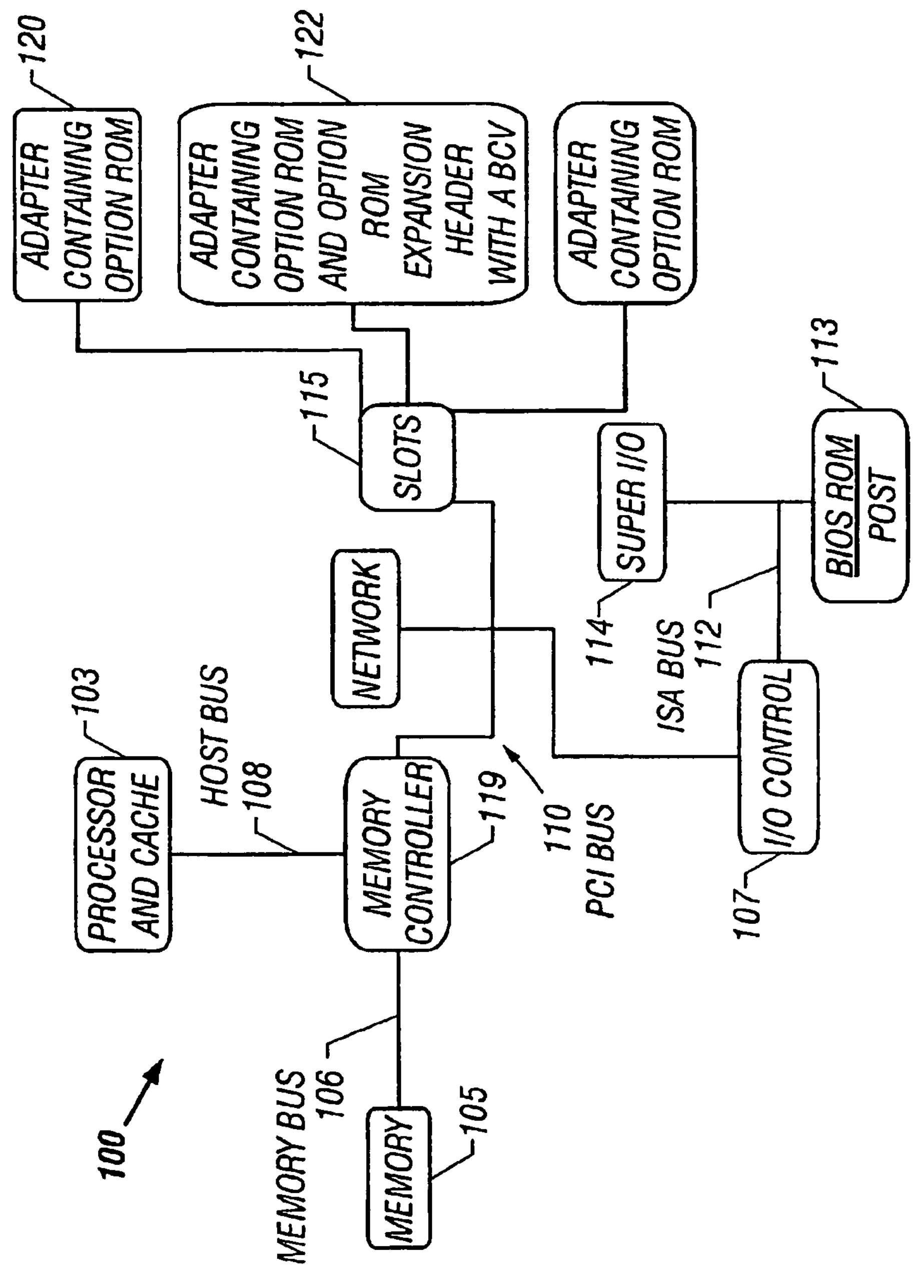

100
PROCESSOR AND CACHE
103
HOST BUS
108
MEMORY CONTROLLER
119
MEMORY BUS
106
MEMORY
105
110
PCI BUS
NETWORK
SLOTS
115
I/O CONTROL
107
ISA BUS
112
SUPER I/O
114
BIOS ROM
POST
113
ADAPTER CONTAINING OPTION ROM
120
ADAPTER CONTAINING OPTION ROM AND OPTION ROM EXPANSION HEADER WITH A BCV
122
ADAPTER CONTAINING OPTION ROM

FIG. 1

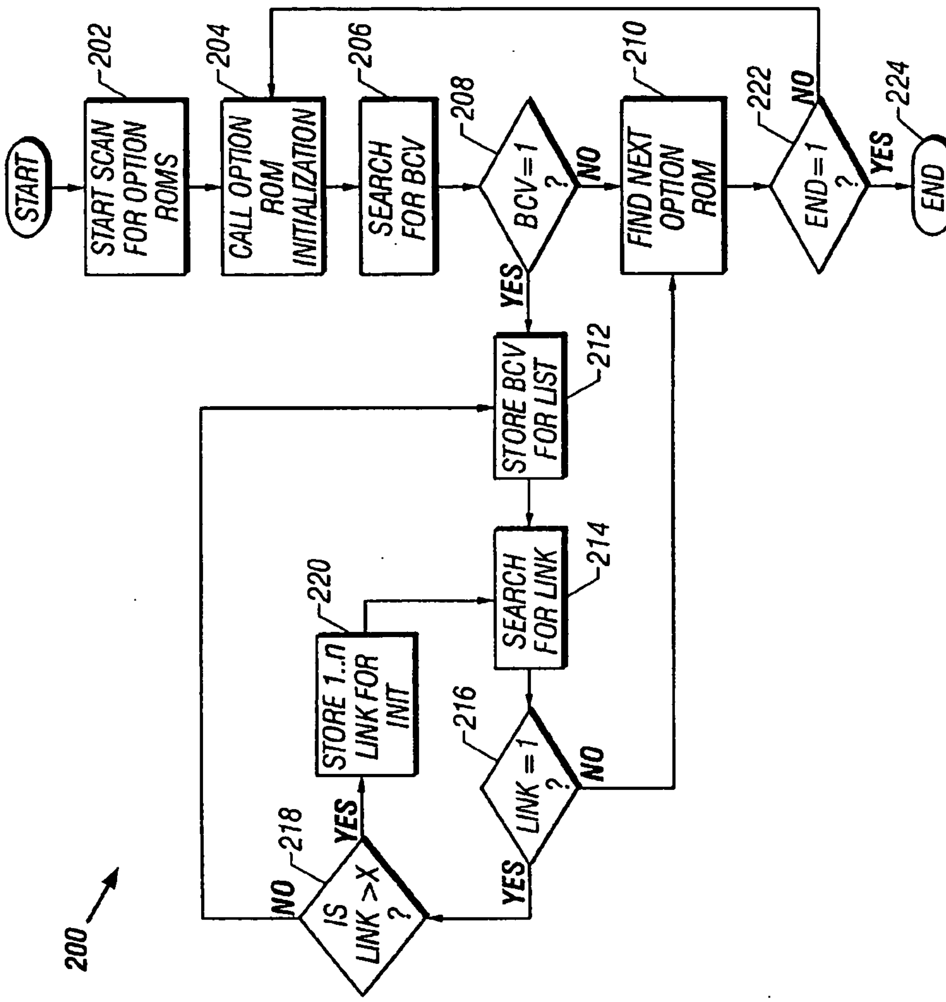

START
START SCAN FOR OPTION ROMS
202
CALL OPTION ROM INITIALIZATION
204
SEARCH FOR BCV
206
BCV=1 ?
208
YES
NO
FIND NEXT OPTION ROM
210
END=1 ?
222
NO
YES
END
224
STORE BCV FOR LIST
212
SEARCH FOR LINK
214
LINK =1 ?
216
YES
NO
IS LINK >X ?
218
YES
NO
STORE 1..n LINK FOR INIT
220
200

FIG. 2

BIOS/UTILITY SETUP DISPLAY

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates in general to computer systems, and more specifically to a BIOS/Utility Setup display program.

2. Description of the Related Art

Personal computer systems have attained widespread use. A personal computer system can usually be defined as a desktop or portable microcomputer that includes a system unit having a system processor or central processing unit (CPU) with associated memory, a display panel, a keyboard, a hard disk storage device or other type of storage media such as a floppy disk drive or a compact disk read only memory (CD ROM) drive. These personal computer systems are information handling systems which are designed primarily to give independent computing power to a single user or group of users.

One of the methods typically used to initialize devices on a personal computer system is called the Basic Input/Output System (BIOS). The BIOS is a program embedded in an integrated circuit component located on the personal computer's main circuit board or mother board. The BIOS runs the Power-On-Self-Test (POST), which is included as part of the BIOS. The POST software initializes the computer hardware so that the computer's operating system can be loaded. All bootable devices must be initialized prior to loading the operating system in order to boot-strap the computer system. Such devices include any Initial Program Load (IPL) devices such as floppy drives or hard drives that can boot-strap and load an operating system.

In order to inform the BIOS of the devices and options installed on a system, computer systems record setup information in a storage system that can be referenced quickly during bootup, typically in nonvolatile system setup memory. A setup program allows the user to configure the operating system and select a particular IPL device. Pressing a particular key on the keyboard during BIOS initialization executes the setup program. Once the setup program executes, the operating system for a computer is loaded from a particular IPL device. A user sets up the particular IPL device through the system Setup program which is part of the BIOS.

The BIOS Boot Specification, Version 1.01, Jan. 11, 1996, ("BIOS Boot Specification") promulgated by "COMPAQ COMPUTER CORPORATION™", "PHOENIX TECHNOLOGIES, LTD™". and "INTEL CORPORATION™", is incorporated herein by reference. The BIOS Boot Specification (BBS) provides a method for the BIOS to identify all IPL devices on a computer system. According to the BBS, the user, through the Setup program, prioritizes the IPL devices so that the system will attempt to boot using each bootable device selected in the order specified. Prioritizing IPL devices is similar to the commonly known boot priority system of attempting to boot using floppy drive A first, then hard drive C. The difference is that the BBS can include additional IPL devices such as a bootable AT Attachment Packet Interface ("ATAPI") CD-ROM drive, a Personal Computer Memory Card Industry Association ("PCMCIA") drive, an embedded network adapter, and "Plug-n-Play" (PnP) devices.

Although the additional IPL devices enable numerous boot options for usage by the user when running the Setup program, the BBS display of bootable devices in the Setup program fails to display all possible bootable drives when

the number of possible bootable devices exceeds the displayable option space limit. For example, if four PowerEdge Raid Controllers™ (PERC) adapter cards are connected in a system holding eight disk pods, such as a Dell Poweredge™ Scalable Disk Storage 100 (SDS 100) unit. In addition, each disk pod in the system contains eight bootable devices, Thus, the user can select from 64 possible bootable devices in theory. The Setup program may be capable of displaying only a fraction of the bootable device selections due to limits in option space. This is a limitation inherent in the BBS standard.

A need has been felt for a Setup Program that is able to display IPL devices as bootable devices according to the BBS standard in a user-friendly manner.

SUMMARY OF THE INVENTION

The present invention accordingly provides a method and computer system for a system setup program that include displaying a subset of bootable devices from an adapter card, such as a PERC card, on a system from which a user may specify a bootable device to serve as a boot drive, and bootable devices to serve as alternative boot drives. The computer system includes a system Basic Input/Output System (BIOS), a system processor, a system memory coupled to the system processor, at least one expansion slot coupled to the system processor via a bus, at least one adapter coupled to the computer system via the at least one expansion slot, and a system BIOS ROM code. The BIOS ROM code detects a plurality of bootable devices on the computer system, selects from the plurality of bootable devices a preselected number of bootable devices for display, and displays the subset of bootable devices on a setup display. The setup display allows a user to specify a bootable device to serve as a boot drive of the computer system.

The method includes detecting a plurality of bootable devices on the computer system, selecting from the plurality of bootable devices a preselected number of bootable devices for display, and displaying the subset of bootable devices on a setup display. The setup display allows a user to specify a bootable device to serve as a boot drive from the computer system.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be better understood, and its numerous objects, features, and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

FIG. 1 is a schematic block diagram of a computer system that is suitable for operating a BIOS/Utility Setup Display according to an embodiment of the present invention.

FIG. 2 is a flowchart showing a method of operating the BIOS/Utility Setup Display of FIG. 1.

The use of the same reference symbols in different drawings indicates similar or identical items.

DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

The following sets forth a detailed description of a mode for carrying out the invention. The description is intended to be illustrative of the invention and should not be taken to be limiting.

The BBS described above and incorporated herein for all purposes as Appendix A, defines the priority of the Boot Connection Vector (BCV). The BCV is a pointer that points

to code for loading the operating system, the boot code, located inside the option ROM. The initialization code in the option ROM performs device initialization, detects whether a peripheral is attached, and optionally hooks INT 13*h*. The BCV resides in a PnP option ROM Expansion Header. An option ROM is located on a given adapter or a device with drives attached. The option ROM stores an initialization code on an option ROM header. A valid option ROM header as defined by the BBS has an Expansion Header with an address within the standard option ROM header. The Expansion Header contains the information used to configure the devices connected through the adapter. The Expansion Header also contains pointers (BCVS) to code in the device's option ROM (the BCV) that BIOS uses to call to boot from the device.

A typical BCV device is a SCSI controller. SCSI controllers are not directly bootable, but by hooking into the BIOS'INT 13*h* services, the SCSI controller drives are added to the system and drive numbers are appended to existing drives. A BCV device is only bootable if a BCV device drive is installed before any other drives in the system. Therefore, control of the order of installation is desirable.

The BCV Priority is specified by a user of INT 13*h* device controllers in a priority list arranged during a setup operation. INT 13*h* is a basic BIOS subroutine that controls part of the configuration of bootable drives. During a Power-On Self-Test (POST), the appropriate INT 13*h* drive support is called for the controllers on the priority list in the order specified by the user. The controllers are listed to control the order of installation of devices using INT 13*h*. Many INT 13*h* devices can be managed by one controller. The user specifically sets the order of priority to account for inconsistencies in booting operations of typical PCs and devices. For example, if an Advanced Technology Attachment (ATA) support that is resident in BIOS is installed before the BIOS option ROM scan, the ATA is first to install into the INT 13*h* services. As a result, bootable drive number **80*h***, commonly known as drive "C", is specified as an ATA drive and devices initialized **81*h*** or higher are rendered non-bootable.

An INT 13*h* device controller installs one or more drives into the BIOS INT 13*h* services by hooking the INT 13*h* software interrupt and chaining to the old vector. By chaining, the INT 13*h* software is connected through a software routine call inserted into the old vector. The software routine call refers the program to the INT 13*h* software, thereby hooking the INT 13*h* software interrupt. The controller can be a PnP Card with a BCV. According to the BBS, the first type of entry in the BCV Table is the BIOS INT 13*h* services for ATA drives. The ATA drive support in the BIOS is to be installable in any order that a user selects; before, in the middle, or after other controllers are installed.

Referring to FIG. **1**, a block diagram shows a computer system that is suitable for operating a BIOS/Utility display routine. Computer system **100** is a server or work station that includes a system processor **103** with cacahe, such as the PENTIUM™ II microprocessor sold by the INTEL™ CORP. Computer system **100** includes an ISA bus **112** coupled to an input/output controller **107** that is often operable coupled to auxiliary devices (not shown), which may include a pointer device, such as a mouse.

The computer system **100** includes a system processor **103** with cache and a memory controller **119** connected to the processor **103** with cache via a local bus or a host bus **108**. A system BIOS **113** is connected to the memory controller **119** via a PCI bus **110** and an ISA bus **112**. Alternatively, instead of an ISA bus, the System BIOS **113** could be coupled to the computer system **100** via a local or host bus. Connected to the PCI Bus **110** are Slots **115**, representing different expansion slots. Controllers for controlling adapter cards are inserted into the expansion slots. The adapter cards are shown as adapters containing option ROMs **120** and adapters containing option ROM Expansion Headers with a BCV **122**.

The computer system **100** executes the BIOS ROM setup code when the system is first powered. A BIOS ROM setup code enables the user to control the order of the drives as defined within the setup program and allows a user to control the order of drives for the operating system to be initialized as a boot drive. In the normal operation of the BIOS ROM setup code, the setup program presents to a user a list of optional adapter selections. However, the space provided in the setup program for displaying bootable device choices may be limited. An illustrative BIOS/Utility Setup Display program controls the number of bootable devices connected to an adapter card that are displayed as options in the BIOS ROM setup code. The program efficiently uses a BIOS ROM setup code space for displaying a limited number of bootable drive selections. For example, the setup code allocates display space among bootable drive selections of multiple adapters. Thus, a particular adapter with a plethora of bootable drive selections does not take up the display area available. The customized BIOS ROM setup code selects a predetermined number of drives from the individual adapter cards to display as bootable drive selections in the option list. The BIOS ROM setup code allows a subset number of bootable drives from the adapters to be displayed and treats remaining bootable drives as active drives that are not controlled as bootable drives.

Referring to FIG. **2**, a flow chart diagram shows a method **200** for selecting the number of bootable devices connected to an adapter card when displaying the bootable devices options in a BIOS/Utility Setup program.

A first operation in the method is scanning for option ROMs **202**. During the scan, a call function is executed in which each adapter with a valid option ROM has associated operating code loaded into a specified section of memory space. During adapter code loading, expansion header information is saved and linked expansion headers are stored into variables for non-controllable devices. Claim 3

A next operation is option ROM initialization **204**, during which a routine jumps to the initial entry point for the option ROM and the code runs the BIOS on the option ROM card. The routine queries the drives that are attached to the card through, for example, a cabling system. The information is returned to the BIOS when finished. A search for BCVs **206** determines the number of BCVs and whether the device is a hard drive device. Another scan is made for option ROMs **210**, followed by a call for option ROM initialization, repeating option ROM initialization **204** followed by the BCV search **206** and another check for a BCV=1 208 until all option ROMs are processed.

If BCV is equal to 1 in logic operation **208**, another BCV is found and the BCV is stored in a list for the controllable devices **212**. Following storing of the BCV, the routine searches for a link **214**, a link search for a pointer for the next BCV. If a link is found, in logic operation **216** a query determines whether enough space is available to increase the number of BCVs in the controllable list **218**. The capacity of the BCV list is determined by the number of spaces for displayed drives and is selected by a user during system BIOS setup. Enforcement of the limitation in displayed BCV

entries in operation **218** prevents the BIOS code from automatically displaying all BCVs connected to an option ROM, thereby filling all available display spaces on the setup screen. For example, if an option ROM has three BCVs and a setup screen has only three spaces available for optional controllable devices, the setup screen capacity is filled. No option ROM or drive attached to the system is accessible by the user for usage as a controllable drive simply because no space is available on the setup screen for displaying the options. The other option ROMs are thus made available only as data drives. The list capacity is a variable in the system BIOS code that provides the number of BCVs per option ROM that is permitted for storage in the list of controllable devices for user selection.

The above description is intended to be illustrative of the invention and should not be taken to be limiting. Other embodiments within the scope of the present invention are possible. Those skilled in the art will readily implement the steps necessary to provide the structures and the methods disclosed herein, and will understand that the process parameters and sequence of steps are given by way of example only and can be varied to achieve the desired structure as well as modifications that are within the scope of the invention. Variations and modifications of the embodiments disclosed herein may be made based on the description set forth herein, without departing from the spirit and scope of the invention as set forth in the following claims.

What is claimed is:

1. A method of operating a computer system having a system Basic Input/Output System (BIOS), the method comprising:

detecting a plurality of bootable devices on the computer system;

selecting from the plurality of bootable devices a subset of a preselected number of bootable devices for display; and

displaying the subset of bootable devices on a setup display from which a user specifies a bootable device to operate as a boot drive of the computer system.

2. The method according to claim **1**, wherein detecting a plurality of bootable devices on the computer system further comprises:

scanning for option ROMs; and

calling option ROM initialization codes.

3. The method according to claim **1**, wherein selecting from the plurality of bootable devices a subset of bootable devices for display further comprises:

searching for at least one Boot Connection Vector (BCV);

storing the at least one BCV in a list; and

comparing the preselected number of bootable devices against the number of BCVs in the list.

4. The method according to claim **3**, wherein comparing the preselected number of bootable devices against the number of BCVs in the list further comprises:

searching for a link;

determining a number of links that are found;

checking the number of links that are found against the preselected number of bootable devices;

if for a link the number of links that are found is greater than the preselected number of bootable devices allocated to the option ROM, storing an identifier of the link found for initialization as a nonbootable device for initialization; and

if the number of links that are found is less than the preselected number of bootable devices allocated to the option ROM, storing an identifier of the link in the list of BCVs.

5. The method according to claim **4**, further comprising:

if the number of links that are found is zero, scanning for an option ROM;

if the number of option ROMS found is zero, ending the method.

6. The method according to claim **1**, wherein:

the plurality of bootable devices includes at least one option ROM, the at least one option ROM coupled to further bootable devices; and

displaying the subset of bootable devices includes displaying the further bootable devices.

7. The method according to claim **1**, further comprising:

classifying a subset of the plurality of bootable devices as active devices independent from control as bootable devices.

8. The method according to claim **1**, wherein the method is performed in a BIOS ROM setup code.

9. The method according to claim **1**, wherein the bootable devices are coupled to at least one adapter card inserted into at least one expansion slot.

10. The method of claim **9**, wherein the adapter card is a Plug and Play (PnP) adapter card with a BCV.

11. The method of claim **1** wherein the plurality of bootable devices consists of initial program load (IPL) devices.

12. A computer system having a system Basic Input/Output System (BIOS), the computer system comprising:

a system processor;

at least one adapter coupled to the computer system; and

a system BIOS ROM code coupled to the system processor, the system BIOS ROM code including a setup code that has limited space for displaying bootable devices preventing the setup code from displaying all possible bootable drives when the number of possible bootable devices exceeds a displayable option space limit, wherein the system BIOS ROM code detects a plurality of bootable devices on the computer system, including bootable devices coupled to one or more adapter cards, selects from the plurality of bootable devices a subset of a preselected number of bootable devices for display from each adapter card, and displays the subset of bootable devices on a setup display from which a user specifies a bootable device to serve as a boot drive of the computer system.

13. The computer system of claim **12**, wherein the system BIOS ROM code scans for option ROMs and calls option ROM initialization codes.

14. The computer system of claim **12**, wherein the system BIOS ROM code that selects from the plurality of bootable devices a subset of bootable devices for display further searches for at least one Boot Connection Vector (BCV), stores the at least one BCV in a list, and compares the preselected number of bootable devices against the number of BCVs in the list.

15. The computer system of claim **14**, wherein the system BIOS ROM code that compares the preselected number of bootable devices against the number of BCVs in the list further searches for a link, determines the number of links found, checks the number of links found against the preselected number of bootable devices, and if the number of links found is greater than the preselected number of bootable devices allocated to the option ROM, stores the link for initialization as a nonbootable device for initialization, and if the number of links found is less than the preselected number of bootable devices allocated to the option ROM, stores the link in the list of BCVs.

16. The computer system of claim 15, wherein the system BIOS ROM code scans for option ROMs if the number of links found is zero, and ends the method if the number of option ROMs found is zero.

17. The computer system of claim 12, wherein:

the plurality of bootable devices includes at least one option ROM, the at least one option ROM coupled to further bootable devices; and

the subset of bootable devices includes the further bootable devices.

18. The computer system of claim 12, wherein a subset of the plurality of bootable devices is classified as active devices independent from control as bootable devices.

19. The computer system of claim 12, wherein the bootable devices are coupled to the one or more adapter cards, the one or more adapter cards being coupled to the computer system via at least one expansion slot.

20. The computer system of claim 19, wherein the adapter card is a Plug and Play (PnP) adapter card with a BCV.

21. The computer system of claim 12 wherein the plurality of bootable devices consists of initial program load (IPL) devices.

22. A computer program product comprising:

a computer readable media including a system Basic Input/Output System (BIOS) routine operable with a setup code that has limited space for displaying bootable devices preventing the setup code from displaying all possible bootable drives when the number of possible bootable devices exceeds a displayable option space limit, the routine including:

detecting a plurality of bootable devices on the computer system, including bootable devices couple to one or more adapter cards;

selecting from the plurality of bootable devices a subset of a preselected number of bootable devices for display from each adapter card; and

displaying the subset of bootable devices on a setup display from which a user specifies a bootable device to operate as a boot drive of the computer system.

23. The computer program product of claim 22 wherein detecting a plurality of bootable devices on the computer system further comprises:

scanning for option ROMs; and

calling option ROM initialization codes.

24. The computer program product of claim 22 wherein selecting from the plurality of bootable devices a subset of bootable devices for display further comprises:

searching for at least one Boot Connection Vector (BCV);

storing the at least one BCV in a list; and

comparing the preselected number of bootable devices against the number of BCVs in the list.

25. The computer program product of claim 22 further comprising: classifying a subset of the plurality of bootable devices as active devices independent from control as bootable devices.

26. The computer program product of claim 24 wherein comparing the preselected number of bootable devices against the number of BCVs in the list further comprises:

searching for a link;

determining a number of links that are found;

checking the number of links that are found against the preselected number of bootable devices;

if for a link the number of links that are found is greater than the preselected number of bootable devices allocated to the option ROM, storing an identifier of the link found for initialization as a nonbootable device for initialization; and

if the number of links that are found is less than the preselected number of bootable devices allocated to the option ROM, storing an identifier of the link in the list of BCVs.

27. The computer program product of claim 21 further comprising:

if the number of links that are found is zero, scanning for an option ROM; and

if the number of option ROMS found is zero, ending the method.

28. The computer program product of claim 22 wherein:

the plurality of bootable devices includes at least one option ROM, the at least one option ROM coupled to further bootable devices; and

displaying the subset of bootable devices includes displaying the further bootable devices.

29. The computer program product of claim 22 wherein the bootable devices are coupled to at least one adapter card inserted into at least one expansion slot.

30. The computer program product of claim 29 wherein the adapter card is a Plug and Play (PnP) adapter card with a BCV.

31. An article of manufacture comprising a computer readable media of signal for implanning a software program operable with a BIOS ROM setup code that has limited space for displaying all possible bootable drives when the number of possible bootable devices exceeds a displayable option space limit including:

means for detecting a plurality of bootable devices on the computer system, including bootable devices coupled to one or more adapter cards;

mean for selecting from the plurality of bootable devices a subset of a preselected number of bootable devices for display from each adapter card; and

user specifies a bootable device to operate as a boot drive of the computer system.

* * * * *

US006513114B1

(12) United States Patent
Wu et al.

(10) Patent No.: **US 6,513,114 B1**
(45) Date of Patent: **Jan. 28, 2003**

(54) **SYSTEM AND METHODS FOR PROVIDING SELECTABLE INITIALIZATION SEQUENCES**

(75) Inventors: **Frank L. Wu**, Austin, TX (US); **Shaojie Li**, Austin, TX (US); **George Mathew**, Austin, TX (US)

(73) Assignee: **Dell Products L.P.**, Round Rock, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/457,016**

(22) Filed: **Dec. 8, 1999**

(51) **Int. Cl.**[7] **G06F 9/445**
(52) **U.S. Cl.** **713/2**
(58) **Field of Search** 713/1, 2

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,388,267 A	2/1995	Chan et al.	....	395/700
5,548,783 A	8/1996	Jones et al.	....	395/836
5,628,027 A	5/1997	Belmont	....	395/821
5,836,013 A	11/1998	Greene et al.	....	395/652
6,353,885 B1 *	3/2002	Herzi et al.	....	713/1

OTHER PUBLICATIONS

U.S. pending patent application Ser. No. 09/237,743 entitled "System and Method for Providing Bios-Level User Configuration of a Computer System" by Jim Dailey, et al. ; Dell USA, L.P., Filed Jan. 26, 1999.

Comdaq Computer Corporation, Phoenix Technologies, Inc., Intel Corporation, "BIOS Boot Specification"; Version 1.01, Jan. 11, 1996.

* cited by examiner

Primary Examiner—Thomas M. Heckler
(74) *Attorney, Agent, or Firm*—Baker Botts L.L.P.

(57) **ABSTRACT**

Methods and system for providing selectable initialization sequences are disclosed. The method of initializing a system includes providing at least one reference associated with a device coupled to the system. A user selects the reference associated with the device and an initialization sequence is determined based upon the selected reference.

16 Claims, 2 Drawing Sheets

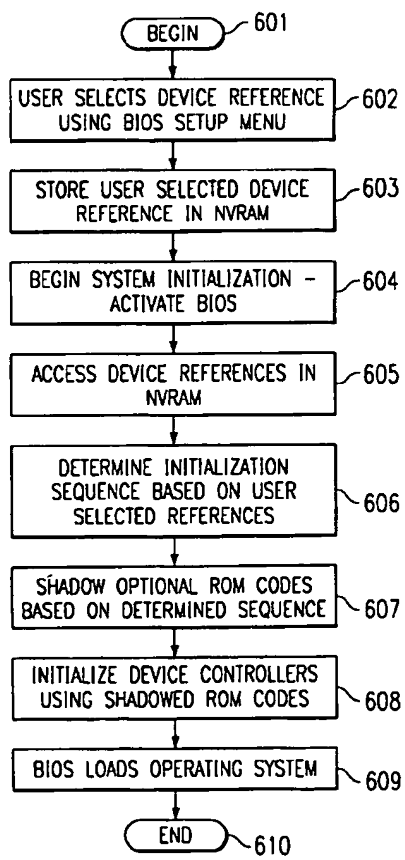
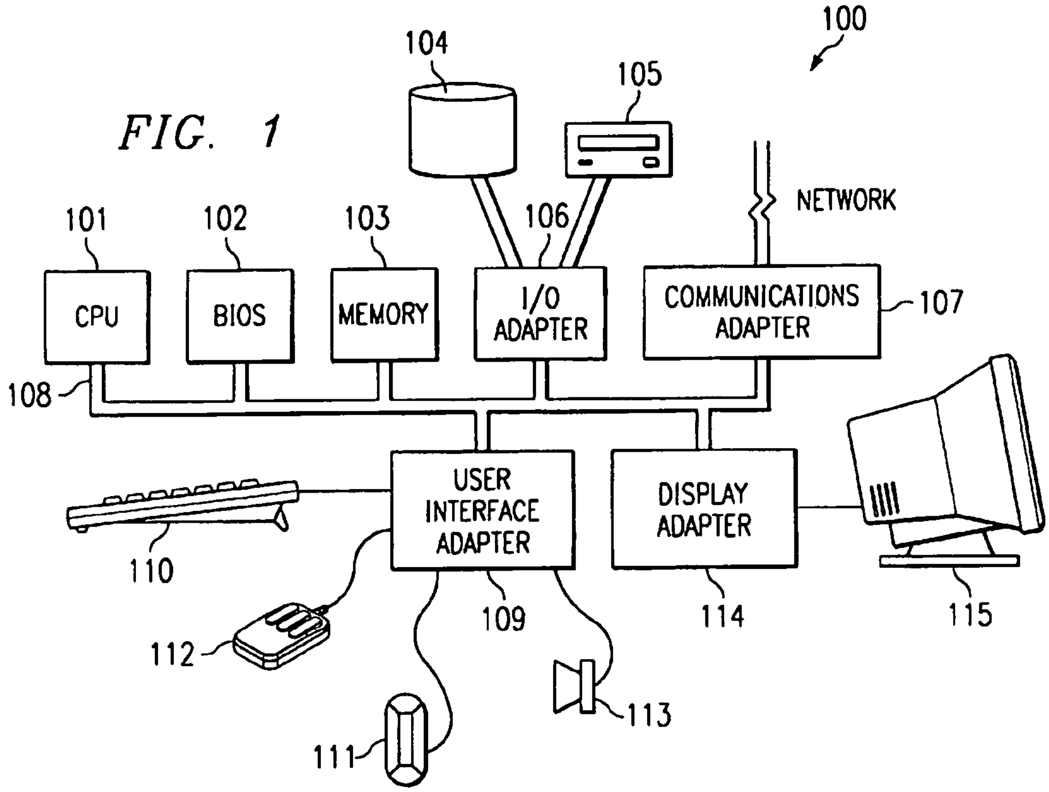


FIG. 1



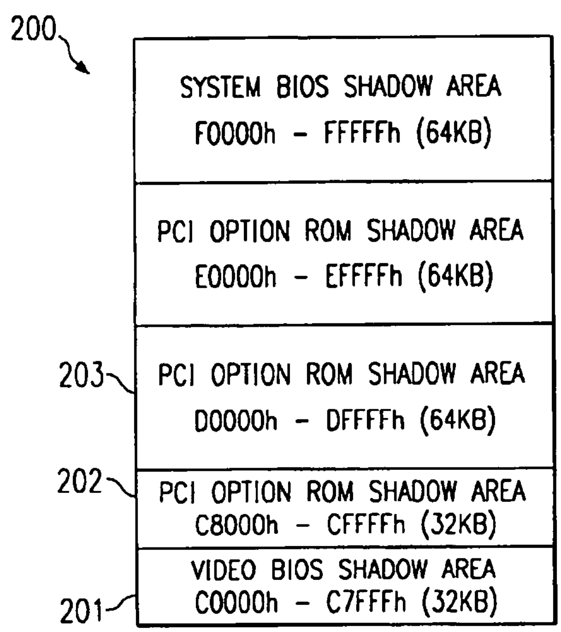


FIG. 2

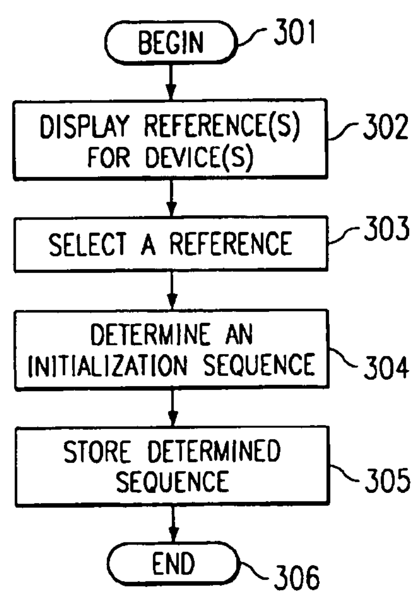


FIG. 3

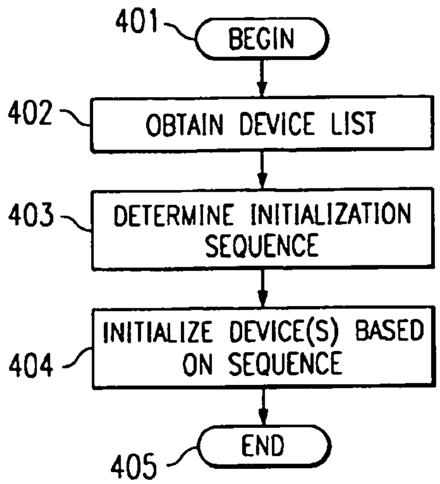


FIG. 4

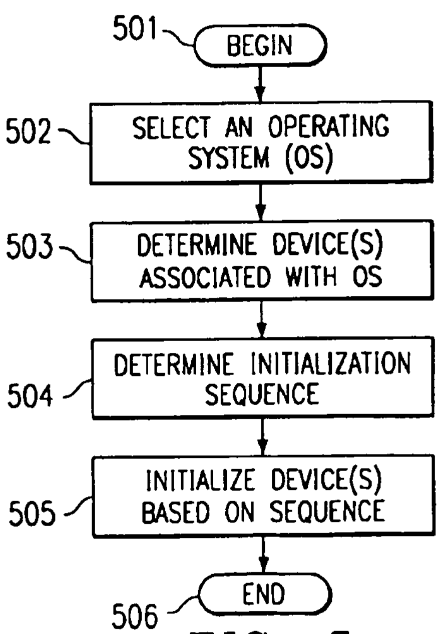


FIG. 5

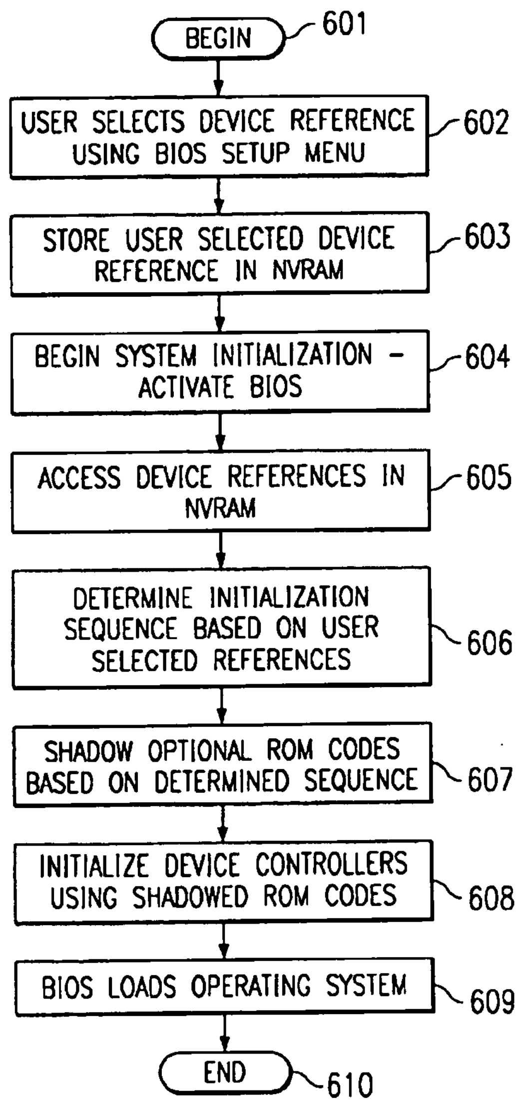


FIG. 6

SYSTEM AND METHODS FOR PROVIDING SELECTABLE INITIALIZATION SEQUENCES

TECHNICAL FIELD

The present disclosure generally relates to initializing computer systems and, more particularly, to a system and methods for providing selectable initialization sequences for computer systems and servers.

BACKGROUND

Technological advancements for increasing the number of peripheral devices for computer systems and servers have taken on many forms. Several types of peripheral devices have become available for computer systems allowing users to expand a system's basic functionality and configuration. For example, conventional systems can include expansion slots or bays connected to the system's mother board for connecting peripheral devices.

Two of the most common interfaces for connecting peripheral devices include Peripheral Component Interconnect (PCI) interface systems and small computer system interface (SCSI) systems. PCI and SCSI interfaces are configured with "hot-plugs" that allow users to add additional peripheral devices such as hard drives, tape drives, CD-ROM drives, scanners, etc. relatively easy. Hot-plugs provide the connection between an interface and a peripheral device using a small opening or panel in the computer system's housing allowing a user to plug in peripheral devices without powering down the computer system.

A PCI interface system is configured to communicate with a system's mother board and includes a plurality of expansion slots spaced closely together for high speed operation. PCI is designed to be synchronized with the clock speed of the mother board's microprocessor and can transmit, up to 64 bits of data using a 188-pin connection. A single peripheral device communicates with the system using a PCI controller that plugs into one of the PCI expansion slots. The controller also has a port connected to one end of the controller card for connecting the peripheral device to the system.

SCSI interface systems are similar to PCI interface systems in that SCSI allows systems to communicate with external peripheral devices. A SCSI interface communicates through a parallel interface that provides quick access and high data transfer rates. For example, a 16-bit interface can provide up to 80 megabytes of data per second. Unlike PCI, a single SCSI can support up to devices when the devices are connected in a "daisy-chain" fashion. One advancement in SCSI technology includes utilizing low voltage differential signaling for communicating with remotely located peripheral devices such as disk drives, scanners, printers, etc.

Conventional systems include peripheral devices connected to PCI or SCSI slots through the use of a device controllers that function as a communication interface between the added peripheral device and the system. Device controllers are configured with optional Read Only Memory (ROM) Integrated Circuits (ICs) that store initialization and run-time code. For example, when a system is turned on or rebooted, the system's initialization routine uses a controller's optional ROM code to initialize the peripheral device and provide run-time code for operating the device. During initialization, the system copies or "shadows" each controller's optional ROM code by loading the code into a portion of available system memory. The system then accesses the shadowed code of each controller during initialization and system operation.

A disadvantage of conventional systems can occur when too many peripheral devices are added to a system. Typically, a finite amount of memory is available to the system during initialization. During the system's initialization routine, the basic input/output system (BIOS) executes a power on self test (POST) that initializes the hardware associated with the system and upon completing POST the BIOS loads an operating system (OS). Therefore, a system having only a finite amount of memory available during POST can initialize a limited number of peripheral devices.

One conventional method used to shadow the controller's optional ROM codes is to shadow the optional ROM codes in a largest-to-smallest sequence. For example, a CD-ROM requiring 64 KB of memory for initialization would have its optional ROM code shadowed before a RAID device that requires 16 KB of memory. This method continues until all of the optional ROM codes are shadowed or until the available memory for shadowing optional ROM codes is exhausted. Consequently, an initialization error could occur if a user wanted to boot an OS from a peripheral device having a controller with a smaller relative optional ROM code that may or may not be shadowed during initialization. If the device having the bootable OS is not initialized, the system will be forced into an undesirable loop causing the system initialization to fail.

Another conventional solution allows users to disable some of the PCI optional ROM codes using an PCI/SCSI configuration menu. Although this technique is effective in "freeing-up" memory, it is disadvantageous because it allows initialization errors if a user unknowingly disables an optional ROM code of a device required for system initialization.

Another solution deployed by conventional systems allows users to disable device controllers via the system's BIOS set-up menu. Though effective, the user must know which device controllers are in which PCI slots before disabling the controller. This solution requires the user to open the system's housing and note the location of each device controller.

SUMMARY

The conceptual groundwork for the present disclosure involves initialization sequences for computer systems. In accordance with the teachings of the present disclosure, a system and methods are described for providing selectable initialization sequences.

According to one aspect of the present embodiment, a method for use in a computer system is disclosed. The method includes providing at least one reference associated with a device coupled to the system, selecting the reference associated with the device, and determining an initialization sequence based upon the selected reference.

More specifically, one embodiment of the method includes storing the initialization sequence in a medium, such as non-volatile memory device, associated with the system.

Another embodiment of the method includes configuring the method to be used with a basic input output system (BIOS) that deploys a power on self test (POST) routine.

A further embodiment of the method includes a POST routine configured to shadow an optional ROM code for the at least one device into a medium coupled to the system, the shadowing being based upon the determined initialization sequence.

According to another aspect of one embodiment, a method of initializing a computer system is disclosed. The method includes providing a device list associated with the system and determining an initialization sequence based upon the device list. The method further includes initializing at least one device coupled to the system based upon the determined sequence wherein the device list includes a reference to at least one device coupled to the system.

In another aspect of one embodiment, a method of providing initialization information for a system is disclosed. The method includes selecting information associated with at least one device coupled to the system and determining an initialization sequence based upon the selected information.

More specifically, one embodiment of the method includes selecting an operating system associated with at least one device coupled to the system.

In a further aspect of one embodiment, a system is disclosed. The system includes at least one processor, at least one storage medium coupled to the processor, at least one device coupled to the system, the device comprising an initialization reference associated with the device. The system further includes a program of instructions, associated with the system, the program of instructions including at least one instruction to determine an initialization sequence of the system based upon the at least one device.

In another aspect of one embodiment, a method of initializing a computer system is disclosed. The method includes providing a device reference for at least one device and allowing a user to select the at least one reference. The method determines an initialization sequence based upon the user selected reference and stores the initialization sequence in a medium associated with the system. The method also includes accessing the stored sequence during a BIOS initialization routine, wherein the BIOS initialization routine deploys a power on self test routine (POST). The method also includes shadowing an optional ROM code for the selected reference wherein the shadowing is based upon the initialization sequence.

It is a technical advantage that an initialization sequence can be selected by a user thereby allowing the user to selectively initialize peripheral devices during a POST routine.

It is another technical advantage that a system is provided having a BIOS set-up menu configured to allow a user to select an initialization sequence for selective peripheral devices. In this manner, the system can initialize a selected peripheral device having a desirable operating system.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present embodiments and advantages thereof may be acquired by referring to the following description taken in conjunction with the accompanying drawings, in which like reference numbers indicate like features, and wherein:

FIG. 1 is a block diagram of a computer system illustrative of embodiment of the present disclosure;

FIG. 2 is an illustration of a shadow memory area for a memory device according to an embodiment of the present disclosure;

FIG. 3 is a flow diagram of one embodiment of a method for determining an initialization sequence;

FIG. 4 a exemplary flow diagram of one embodiment of a method for initializing a system;

FIG. 5 illustrates a flow diagram of one embodiment of a method of initializing a system upon selecting information associated with the system; and

FIG. 6 illustrates a flow diagram of one embodiment of a method of initializing a system based on a selected initialization sequence.

DETAILED DESCRIPTION

In an advantageous embodiment of the disclosure a computer system displays references for one or more devices coupled to the system. The references are displayed to a user via a set-up menu and allows a user to select a desirable system initialization sequence. The selected sequence is stored in a memory device, such as a non-volatile memory device, coupled to the system. During initialization of the system, the system's BIOS deploys a power on self test (POST) routine that copies or "shadows" initialization information for each device according to the selected sequence. In this manner, a customizable initialization sequence is afforded ensuring selective devices coupled to the system are initialized during a POST routine. Therefore, the disadvantages of current systems are overcome by allowing a user to load or boot an operating system from a desirable device. The system advantageously avoids encountering boot errors incurred during initialization due to non-initialized devices having a desirable operating system.

Referring to FIG. 1, a block diagram of a computer system illustrative of one embodiment of the present disclosure is shown. A system, indicated generally at **100**, includes a central processing unit (CPU) **101** connected via at least one bus **108** to a basic input output system (BIOS) firmware **102**, and memory, such as RAM, ROM, EEPROM, and any other memory devices, collectively designated by reference numeral **103**. System **100** further includes an input/output adapter **106** for connecting peripheral devices such as SCSI drives **104** and RAID drives **105**, and a display adapter **114** for connecting a display device **115** such as a Flat Panel Display (FPD) or a Cathode Ray Tube (CRT). A user interface adapter **109** is provided for connecting a keyboard **110**, a mouse **112**, a speaker **113**, a microphone **111**, and/or other user interface devices such as game controllers, touch pads, etc. System **100** also includes a communications adapter **107** for connecting system **100** to an information network such as an Intranet or the Internet.

BIOS firmware **102** includes a built-in software program, referenced generally as BIOS, accessible to system **100**. The BIOS includes instructions for controlling the system's devices and testing memory. During operation, when system **100** is initially powered up, CPU **101** activates the BIOS. The BIOS runs a series of tests using a power on self test (POST) routine that initializes the system's hardware, chip registers, disk drives, power management, I/O ports and any other device requiring initialization by a POST routine. During POST, initialization information associated with the system's components are copied or "shadowed" in a predetermined sequence into a shadow area of the system's memory. For example, system **100** shadows an optional ROM code for RAID drive **105** to an available memory location within memory **103** based upon the predetermined sequence. Therefore, initialization of RAID device **105** is ensured during the POST routine such that, in one embodiment, an operating system stored within RAID device **105** can be loaded upon completing the POST routine.

FIG. 2 is an illustration of a shadow memory area for a memory device according to an embodiment of the present disclosure. A shadow memory area **200** includes memory location addresses for a plurality of devices. In an exemplary form, one embodiment illustrated includes the video BIOS

for system 100 being shadowed in a first location as referenced by 201 followed by a PCI option ROM code of peripheral device, such as RAID drive 105, being shadowed second in a second location as referenced by 202. Subsequent locations for additional devices are also illustrated for shadow memory area 200 thereby depicting the total shadow memory available for system 100.

The present disclosure advantageously allows a user to select a sequence for shadowing initialization information into shadow memory area 200 of system 100 during the POST routine. The selected sequence allots a range of available memory addresses based on the determined initialization sequence. In the example illustrated in FIG. 2, the VIDEO BIOS for system 100 was determined to be the first device in the initialization sequence. Therefore, system 100 shadows the VIDEO BIOS at the lowest available shadow memory location (i.e. C000h-C7FFFh). Subsequently, first PCI Option ROM code 202 was shadowed second reflecting the second device determined in the initialization sequence. First Option ROM 202 occupies the next available shadow memory location (i.e. C8000h-CFFFh). Therefore, initialization information for each device is shadowed based upon the determined initialization sequence.

In this manner, devices such as RAID drives, SCSI drives, tape drives, etc. operably coupled to the system can be shadowed and initialized in a selectable sequence. Through providing a selectable initialization sequence, and in particular a selectable shadowing sequence, peripheral devices storing desirable information and applications can be selectively initialized allowing device accessibility during and after POST.

Referring now to FIG. 3 a flow diagram of one embodiment of a method for determining an initialization sequence is shown. The method can be configured to be used with system 100 as illustrated in FIG. 1 or any other system configured to use initialization sequences.

In operation the method begins at step 301. At step 302, references for a plurality of devices coupled to system 100 are provided to a user through a user interface. For example, the user interface could include references for a CD-ROM device, a floppy disk drive, RAID devices, and/or a tape drive coupled to the system. In a preferred embodiment, the references are provided using an initialization configuration menu similar to a BIOS "set-up" menu commonly known in the art and accessible by system users. The set-up menu displays the current initialization sequence and references for the devices coupled to the system.

The user interface allows a user to modify the initialization sequence by selecting a device reference. For example, the user may want to initialize RAID drive 105 before any other storage device coupled to system 100. Therefore, the user would select that device to be initialized before the other storage devices. Subsequently, the method proceeds to step 304 and determines the initialization sequence based upon the user's selection. The method then proceeds to step 305 where the initialization sequence is stored memory 103 of system 100.

Upon initialization of system 100, the stored initialization sequence is accessed and during POST, initialization information is copied or shadowed memory 103 of system 100. For example, if RAID drive 105 was selected to be initialized first, the controller's PCI optional ROM code for RAID drive 105 is shadowed first, thereby ensuring that the RAID drive 105 is initialized and accessible by system 100 during and after POST.

In an alternate embodiment of method 300, a user can select the initialization of all of the devices displayed in the set-up menu. One skilled in the art can appreciate that method 300 can be modified to repeat steps 303 and 304 allowing a user to select the sequence for all of the devices displayed within the set-up menu.

Referring now to FIG. 4 an exemplary flow diagram of one embodiment of a method for initializing a system is shown. The method can be configured to be used with system 100 as illustrated in FIG. 1 or any other system configured to use initialization sequences.

During operation, the method begins at step 401. At step 402 the method accesses a device list stored within memory 103 of system 100. In a preferred embodiment, the device list is stored in non-volatile memory and is accessed by the system's BIOS during initialization. The device list includes references to the system's devices (i.e. hard drives, tape drive, ROM drives, etc.) and an associated initialization "priority" for each device. For example, a RAID drive 105 may have a higher initialization priority than SCSI drive 104.

Upon accessing the device list, the method proceeds to step 403 where an initialization sequence is determined based upon the accessed information within the device list. For example, if RAID drive 105 has a higher priority than SCSI drive 104, RAID drive 105 would be initialized first. The method would then proceed to step 404 where system 100 would initialize the devices based upon the determined initialization sequence. Initialization is based upon the determined sequence and includes shadowing or copying initialization information into a shadow memory area of system 100. Similar to the illustration of FIG. 2, the highest priority initialization information is shadowed in the first available memory location with the next highest being shadowed after the first, etc. In this manner, a device can be initialized by the BIOS during POST based upon the information provided in the device list and the determined initialization sequence.

Referring now to FIG. 5 an illustration of a flow diagram of one embodiment of a method of initializing a system upon selecting information associated with the system is shown. The method illustrated in FIG. 5 can be configured to be used with system 100 as illustrated in FIG. 1 or any other system configured to use initialization sequences.

During operation the method begins at step 501. At step 502, a user selects information associated with system 100. For example, a user may want to select one or more operating system's, such as LINUX or Windows NT, to boot from during the system's initialization. Upon the user selecting the information, the method proceeds to step 503 where the method determines which device is associated with the selected information. For example, the method would determine the storage location of an selected operating system such as SCSI drive 104. The method then proceeds to step 504 where an initialization sequence is determined based upon the selected information and at step 505 the method initializes the devices based upon the determined sequence.

As noted above, when system 100 is rebooted or turned on, the system's BIOS would access the determined initialization sequence. Upon deploying a POST routine, system 100 would copy or shadow initialization information, such as an optional ROM code for a RAID controller, into the system's shadow memory location. In this manner, initialization of the device during POST and having a desirable operating system or application is ensured.

Referring now to FIG. 6, an illustration a flow diagram of one embodiment of a method of initializing a system based on a selected initialization sequence is shown. The method illustrated in FIG. 6 can be configured to be used with

system 100 as illustrated in FIG. 1 or any other system configured to use initialization sequences.

During operation the method begins at step 601. At step 602 a user selects a device reference using a BIOS set-up menu associated with system 100. For example, a user may want to initialize SCSI drive 105 coupled to system 100 and storing an operating system such as LINUX. The user would select the device reference for SCSI drive 105 and the method would proceed to step 603 where the user selected reference is stored in a non-volatile memory device, NVRAM, coupled to system 100. In a preferred embodiment, the user selected reference is stored in a device list located within the NVRAM. The device list includes initialization parameters associated with the devices coupled to system 100.

Upon storing the user selected device reference, the method proceeds to step 604 where the system initialization begins. Initialization includes activating the system's BIOS and deploying a POST routine. The method proceeds to step 605 where the device list stored in NVRAM is accessed. At step 606 information within the accessed device list is used to determine an initialization sequence for the devices coupled to system 100. The initialization sequence is determined by the user selected device references accessed in step 605 and stored within the device list.

Upon determining the initialization sequence, the method proceeds to step 607 where the optional ROM code for the selected device is copied or "shadowed" into the available shadow memory area of system 100. The initialization information is shadowed in the first available address of the shadow memory area. As previously illustrated, FIG. 2 depicts a shadow memory area for several devices. The user selected initialization sequence determines which address locations within shadow memory area are used during the POST routine. FIG. 2 illustrates the system's video BIOS ROM code shadowed into memory 103 followed by a first PCI optional ROM code, and a second PCI optional ROM code, etc. Therefore, the user selected initialization sequence determines the memory location for shadowing a selected device's initialization code. In this manner, a user can ensure a devices initialization information is shadowed thereby allowing access to the device during and after the POST routine.

Upon shadowing the optional ROM codes for the the shadowed codes are used to initialize the system's devices and device controllers. The method advantageously allows the system to initialize controllers based upon the user selected initialization sequence. The user can select a storage device having an desirable operating system to be initialized during POST allowing the BIOS to load an OS from the selected storage device. Upon the system loading an operating system the method proceeds to step 610 where the method ends.

Although the disclosed embodiments have been described in detail, it should be understood that various changes, substitutions and alterations can be made to the embodiments without departing from their spirit and scope.

What is claimed is:

1. A method of initializing a system comprising:

providing a device reference for at least one device;

allowing a user to select the at least one reference;

determining an initialization sequence based upon the user selected reference;

storing the initialization sequence in a medium associated with the system;

accessing the stored sequence during a BIOS initialization routine, wherein the BIOS initialization routine deploys a power on self test routine (POST);

shadowing an optional ROM code for the selected reference; and

wherein the shadowing is based upon the initialization sequence.

2. A method according to claim 1 further comprising:

after the operation of shadowing an optional ROM code for the selected reference, loading an operating system from the device associated with the selected reference.

3. A method according to claim 1 further comprising:

after the operation of shadowing an optional ROM code for the selected reference, using the shadowed optional ROM code to initialize the device associated with the selected reference; and

after the operation of using the shadowed optional ROM code to initialize the device associated with the selected reference, loading an operating system from the device associated with the selected reference.

4. A method according to claim 3 wherein:

the operation of shadowing an optional ROM code for the selected reference comprises copying the optional ROM code for the selected reference from a non-volatile memory to a random access memory (RAM); and

the operation of loading an operating system from the device comprises copying at least part of the operating system from a disk drive to the RAM.

5. A method according to claim 1 for initializing a system with multiple devices, wherein:

the operation of determining an initialization sequence based upon the user selected reference comprises determining an initialization sequence that gives the device associated with the user selected reference priority over other devices among the multiple devices; and

the operation of shadowing an optional ROM code for the selected reference, based upon the initialization sequence, comprises shadowing devices according to the stored initialization sequence, such that the device associated with the user selected reference is shadowed before the other devices are shadowed.

6. A method for managing at least part of a boot process in a computer system with a processor, at least one storage medium, at least first and second devices, a first optional ROM code associated with the first device, and a second optional ROM code associated with the second device, the method comprising:

receiving input from a user specifying a higher initialization priority for a device reference associated with one of the first and second devices;

storing, in the at least one storage medium, an initialization sequence for the first and second devices, such that the initialization sequence corresponds to the user-specified initialization priority;

accessing the stored initialization sequence during a basic input/output system (BIOS) initialization routine;

shadowing the first optional ROM code before shadowing the second optional ROM code if the stored initialization sequence reflects a higher user-specified initialization priority for the first device; and

shadowing the second optional ROM code before shadowing the first optional ROM code if the stored initialization sequence reflects a higher user-specified initialization priority for the second device.

7. A method according to claim 6 further comprising a subsequent operation of attempting to load an operating system from at least one of the first and second devices.

8. A method according to claim 6 further including subsequent operations comprising:

using the shadowed optional ROM code to initialize at least one of the first and second devices; and

subsequently, attempting to load an operating system from at least one of the first and second devices.

9. A program product for managing at least part of a boot process in a computer system with a processor, at least first and second devices, a first optional ROM code associated with the first device, and a second optional ROM code associated with the second device, the program product comprising:

at least one storage medium; and

computer instructions stored in the at least one storage medium, wherein the computer instructions, when executed, perform operations comprising:

receiving input from a user specifying a higher initialization priority for a device reference associated with one of the first and second devices;

storing, in the at least one storage medium, an initialization sequence for the first and second devices, such that the initialization sequence corresponds to the user-specified initialization priority;

accessing the stored initialization sequence during a basic input/output system (BIOS) initialization routine;

shadowing the first optional ROM code before shadowing the second optional ROM code if the stored initialization sequence reflects a higher user-specified initialization priority for the first device; and

shadowing the second optional ROM code before shadowing the first optional ROM code if the stored initialization sequence reflects a higher user-specified initialization priority for the second device.

10. A program product according to claim 9, wherein the computer instructions perform further operations in a subsequent stage of the boot process, the further operations comprising the operation of attempting to load an operating system from at least one of the first and second devices.

11. A program product according to claim 9, wherein the computer instructions perform further operations in subsequent stages of the boot process, the further operations comprising:

using the shadowed optional ROM code to initialize at least one of the first and second devices; and

subsequently, attempting to load an operating system from at least one of the first and second devices.

12. A system comprising:

at least one processor;

at least one storage medium coupled to the processor;

at least first and second devices coupled to the processor;

a first optional ROM code associated with the first device;

a second optional ROM code associated with the second device; and

computer instructions stored in the at least one storage medium, wherein the computer instructions, when executed, manage at least part of a boot process in the system by performing operations comprising:

receiving input from a user specifying a higher initialization priority for a device reference associated with one of the first and second devices;

storing, in the at least one storage medium, an initialization sequence for the first and second devices, such that the initialization sequence corresponds to the user-specified initialization priority;

accessing the stored initialization sequence during a basic input/output system (BIOS) initialization routine;

shadowing the first option ROM code before shadowing the second optional ROM code if the stored initialization sequence reflects a higher user-specified initialization priority for the first device; and

shadowing the second option ROM code before shadowing the first optional ROM code if the stored initialization sequence reflects a higher user-specified initialization priority for the second device.

13. A system according to claim 12, wherein the computer instructions perform further operations in a subsequent stage of the boot process, the further operations comprising the operation of attempting to load an operating system from at least one of the first and second devices.

14. A system according to claim 12, wherein the computer instructions perform further operations in subsequent stages of the boot process, the further operations comprising:

using the shadowed optional ROM code to initialize at least one of the first and second devices; and

subsequently, attempting to load an operating system from at least one of the first and second devices.

15. A system according to claim 12, wherein the system comprises a personal computer.

16. A system according to claim 12, wherein the system is configured as a server.

* * * * *